a站项目文档

项目小组 xx小组

小组成员 宁志豪、何玉洁、刘苏云、代贝妮、吴华、陈海洪、叶涵予

联系方式

重庆师范大学软件工程系

摘要

面对综合视频平台内容泛娱乐化、垂直领域用户缺乏高质量互动与社区归属感的行业背景，本项目旨在为 16-28 岁 Z 世代的兴趣群体打造一个专注于二次元、科技等领域的垂直视频社区——A 站。项目目标是通过构建“高质量视频+实时弹幕+深度社群”的核心模式，解决内容消费者“找同好难”与创作者“分享渠道少”的双重痛点。

开发过程采用分阶段迭代策略：首阶段聚焦 PC 端，完成视频上传播放、弹幕互动、用户中心等最小可行产品；后续阶段逐步扩展社区功能与移动端。技术栈上，选用 C++/QML、JavaScript 与 MySQL 等成熟技术，确保开发效率与系统稳定性。

项目成果预期为一个具备独特社区文化、高用户粘性的兴趣平台，为内容生态与商业化奠定基础，具备较好的市场可行性与发展前景。

关键词：垂直视频社区，弹幕互动，Z 世代，兴趣社群，迭代开发

日期	修改	描述	作者
2025 年 9 月 5 号	0.1.0	初始版本，明确基本目标	
2025 年 10 月 16 号	0.1.2	进一步完善需求文档	
2025 年 10 月 19 号	0.2	对需求文档进行大改：愿景文档、需求金字塔、涉众分析、用况建模、基本流、备选流	
2025 年 10 月 30 日	0.2.1	更新文档结构	
2025 年 11 月 23 日	0.2.2	对用例进行大的修改	
2026 年 1 月 9 日	0.3	修改用况，架构和 uml 图只完成一个完整基本流	

目录

1 立项

1.1. 项目起源与提案

1.1.1 解决需求与痛点

外部需求：市场上综合视频平台内容泛娱乐化，垂直领域（二次元、硬核科技等）爱好者和深度创作者缺乏高质量互动环境与同好圈子，存在“社区归属感弱”的痛点。

用户痛点：内容消费者难以获得垂直领域高质量内容与实时互动体验，内容创作者缺少便捷分享渠道与公平流量机制。

1.1.2 捕捉机会

市场驱动：互联网视频内容消费持续增长，Z 世代及年轻用户（16-28 岁）对“深度互动+兴趣社区”需求旺盛，存在未被满足的细分市场空白。

战略驱动：布局视频社区赛道，通过“核心兴趣模块”构建差异化竞争力，为未来拓展直播、电商、游戏等商业化模块奠定基础。

1.1.3 商业价值驱动

增收：短期通过基础功能积累用户，长期可通过会员、直播打赏、创作者激励等模块直接创造收入，同时提升用户留存间接促进商业价值增长。

增效：通过自动化内容审核、云服务器存储与管理，替代部分人工操作，降低平台运营成本；借助协同工具提升团队开发与运营效率。

1.2. 项目提案

1.2.1 发起方

由我们七人团队发起。

1.2.2 提案形式

结合“business case”，明确项目愿景、目标、功能模块与实施路径，而非单一原型。

1.2.3 核心内容

1.2.3.1 解决的问题：

垂直领域用户“缺高质量内容、缺互动、缺归属感”，创作者“缺工具、缺流量、缺反馈”的双重痛点。

1.2.3.2 目标用户

核心为 16-28 岁学生与年轻上班族（内容消费者）、平台生态基石的内容创作者（UP 主）。

1.2.3.3 解决方案

打造以“兴趣模块”为核心的视频社区，核心功能涵盖视频上传/播放/弹幕互动、创作者后台、用户中心，分三阶段落地（首阶段聚焦电脑端核心功能，后续拓展手机端）。

1.2.3.4 时机

当前视频内容消费增长，细分市场需求未被满足，先发布局电脑端可快速验证需求，抢占“垂直社区”赛道先机。

1.2.3.5 预期商业价值

短期积累核心用户与创作者，中期通过会员、直播等实现增收，长期构建“内容+社区+商业化”生态，提升市场份额。

1.2.3.6 初步资源评估

技术栈确定为后端 C++/QML/JavaScript、前端 QML、云数据库 MySQL、云服务器、云存储；人力为 7 人团队，分阶段投入开发，首阶段聚焦最小可行产品（电脑端核心功能）。

1.3. 项目可行性分析

1.3.1 市场可行性

目标用户（Z 世代）对弹幕互动、兴趣社区接受度高，且细分市场竞争压力较小，需求真实存在；电脑端用户基数稳定，先开发电脑端可覆盖核心办公、居家使用场景。

1.3.2 选定的技术栈

（C++、MySQL、Socket 等）成熟，电脑端核心功能（视频播放、弹幕、基础审核）无技术壁垒，可依托云服务器实现快速部署。

1.3.3 经济可行性

首阶段仅开发电脑端，暂不涉及手机端与高级算法，成本可控；长期可通过会员、直播等模块实现盈利，ROI（投资回报率）具备增长潜力。

1.3.4 运营可行性

审核员、管理员角色分工明确，基础审核流程与用户运营机制清晰，可支撑电脑端初期平台运转。

1.4. 项目 Business Case

1.4.1 核心定位

1.4.2 关键组成部分

1.4.2.1 业务建模

现状分析：用流程图呈现当前“垂直领域用户分散、创作者分享难”的痛点，例如“用户找垂直内容→浏览泛娱乐平台→无法精准匹配”的低效流程，可视化痛点所在；同时体现“电脑端仍是部分用户（如学生、办公族）观看长视频、创作内容的主要场景，却缺乏专属垂直社区”的现状缺口。

未来设计：通过新业务模型图展示“用户（电脑端）→兴趣频道→精准内容→弹幕互动→关注创作者”的高效流程，以及“创作者（电脑端）→便捷上传→审核→流量分发→粉丝互动”的闭环，直观呈现解决方案。

工具应用：基于“商业模式画布”9 大模块构建逻辑，例如“客户细分”为 16-28 岁垂直领域用户与创作者，“价值主张”为“电脑端优质体验+弹幕互动+兴趣社区+创作者友好工具”，“收入来源”短期为会员，长期拓展直播打赏等。

1.4.2.2 详细财务分析

成本：首阶段（电脑端开发）云服务器租赁、人力开发成本；中期（手机端迭代）额外开发与适配成本；长期（功能拓展）运营与维护成本。

收益：量化会员付费、直播打赏等收入预期，计算ROI与投资回收期，例如“预计电脑端上线1年用户达x万，会员转化率x%，单会员年费x元，年会员收入x万；手机端上线后用户规模预计提升x%，收入同步增长”。

1.4.2.3 全面风险评估

风险类型：技术风险（弹幕屏蔽算法实现难度）、市场风险（电脑端用户留存不及预期）、合规风险（内容审核疏漏）、迭代风险（电脑端与手机端数据同步问题）。

缓解策略：技术风险通过“先基础屏蔽功能，后迭代算法”解决；市场风险通过“聚焦核心垂直领域+电脑端专属活动”提升粘性；合规风险通过“人工审核+关键词自动检测”规避；迭代风险通过“统一数据接口设计”确保多端数据互通。

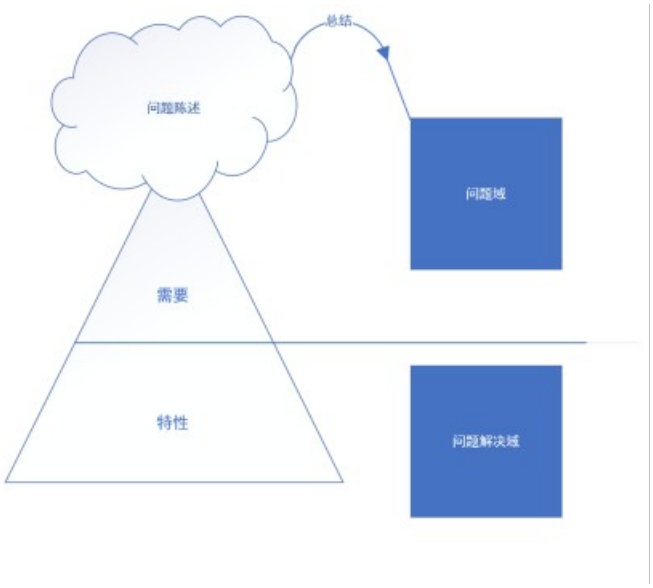
1.4.2.4 实施计划

时间线：首阶段（8个月-1年）完成电脑端核心功能（视频上传/播放、弹幕、基础审核、用户中心）；第二阶段上线电脑端社区系统（评论、关注、动态）；第三阶段开发手机端，同步推进付费与直播功能。

资源需求：首阶段需7人团队（开发、测试、运营），云服务器（腾讯云），MySQL云数据库，云存储等；手机端开发阶段需额外补充移动端开发人力。

2 愿景

2.1. 问题陈述与需求金字塔



2.1.1 问题一

要素	描述
问题	内容消费者缺乏能进行高质量欣赏和实时互动、方便联系同好的平台。
影响	难以在社区中找到契合自身兴趣的高质量内容与互动场景，无法充分满足文化消费需求，对社区的参与感和融入感较低。
结果	可能逐渐减少在该社区的停留时间，甚至放弃使用该社区，转而寻找其他能更好满足需求的平台，导致社区用户流失。
优点	用户能在社区中获得丰富、优质且符合兴趣的内容，通过实时互动与同好深入交流，增强对社区的认同感和归属感，提高用户粘性和活跃度，促进社区文化的繁荣发展。

2.1.2 问题二

要素	描述
问题	内容创造者缺少社区对应的分享平台来分享个人见解或展示自己、寻找同好的渠道。
影响	创作内容缺乏有效的展示和传播途径，难以吸引目标受众，创作动力可能受挫，且难以与同领域创作者交流合作，限制创作水平的提升。
结果	部分创作者可能因缺乏发展机会而减少创作频率或放弃在该社区创作，导致社区优质内容供给不足，内容多样性受限，影响社区的整体吸引力。

优点	创作者拥有专属且高效的分享平台，能够精准触达目标受众，获得更多反馈和支持，激发创作热情，提升创作质量。同时，便于与同好交流，促进创作思路的拓展和创新，为社区带来丰富多样的优质内容。

2.1.3 问题三

要素	描述
问题	平台审核员面对海量用户生成内容，缺乏高效、精准且标准统一的工具与方法来确保内容安全与合规。
影响	审核效率低下，导致违规内容不能及时处理，影响社区内容质量；尺度把握不一，可能造成对合规内容的误判或对违规内容的放纵，引发用户不满，损害社区公平性和公信力。
结果	审核员工作压力大，可能出现职业倦怠，导致审核团队不稳定；社区内容安全问题频发，影响用户体验，降低用户对社区的信任度，甚至可能面临监管风险。
优点	审核员借助高效、精准且标准统一的工具与方法，能够快速准确地处理海量内容，提高审核效率和质量，确保内容安全与合规。减轻工作压力，提升工作满意度和审核团队的稳定性，维护社区的良好秩序和公平环境，增强用户对社区的信任。

2.1.4 问题四

要素	描述
问题	系统管理员支撑高并发、强交互的视频社区平台时，缺乏对系统全局健康状况、用户权限和数据资源的集中、可视化管理能力。
影响	难以实时掌握系统运行状态，不能及时发现和解决系统故障，影响平台的稳定性和可用性；用户权限管理混乱，可能导致数据泄露、非法操作等安全问题；数据资源管理分散，不利于数据的整合和利用，影响平台的决策和发展。
结果	平台频繁出现故障，影响用户体验，导致用户流失；安全事件频发，损害平台声誉，面临法律风险和经济损失；数据资源无法有效利用，影响平台的业务创新和竞争力提升。
优点	系统管理员通过集中、可视化的管理界面，能够实时监控系统全局健康状况，及时发现并解决系统问题，确保平台的稳定运行。精准管理用户权限，保障数据安全，防止安全事件发生。有效整合和利用数据资源，为平台的决策和发展提供有力支持，提升平台的竞争力和可持续发展能力。

2.2. 涉众与用户

2.2.1 涉众

涉众	涉众类型	简要描述

视频消费与互动者（观看视频的人）	Users(用户们） ->主要服务对象，系统的主要使用者	获取高质量、感兴趣的内容（如动画、游戏、知识等）；流畅的播放与弹幕互动体验；个性化的内容推荐；强烈的社区归属感
Up 主（内容创作者）		便捷的视频上传、管理和数据分析工具；公平的流量分发与变现机制（如创作激励）；与粉丝的有效互动渠道；良好的社区氛围以支持持续创作
审核员/内部运营人员/		高效的内容审核工具与明确的规范，以维护社区氛围与安全；丰富的后台数据以支持运营决策
投资人/股东/甲方（假定，实际上没有）	Sponsors(发起人）->我们七人	他们往往关注视频平台的用户增长情况，市场份额盈利情况和商业回报等内容
业务经理这类人		分析平台在我们视频行业的独特定位和优势，从而指导开发出更适应市场的平台
包括开发，测试，运维->我们七人	Developers(开发人员）	我们关注的是一个清晰，稳定的产品需求和技术方案；合理的愿景开发周期；以及可维护可扩展的系统架构
内部监管机构（审核员）	Authorities(权威)	平台可以审核通关的内容应该符合法律法规，特别是版权和青少年保护什么的
领域专家（假定，实际上多半没有，对于社区文化的研究者）		他负责保证我们项目可以保持独特的社区文化模块（比如二次元社区，弹幕文化等），用于体现我们项目的核心价值
消费者（购买大会员或者推流的人，或者买直播打赏道具的人->暂定）	Customers(客户）	他负责保证我们项目可以保持独特的社区文化模块（比如二次元社区，弹幕文化等），用于体现我们项目的核心价值

2.2.2 用户角色与画像

角色	核心特征与目标	核心需求

普通用户	视频消费与互动者	1. 流畅地搜索和发现感兴趣的视频。 2. 通过弹幕、评论、点赞与他人互动。 3. 关注 UP 主，形成自己的信息流。 4. 管理自己的收藏夹和观看历史
UP 主	内容创作者，平台生态核心，已注册用户的泛化	1. 便捷地上传、管理和发布视频。 2. 获得清晰的数据反馈（播放量、粉丝数等）。 3. 与粉丝进行有效互动（评论、动态）
审核员	内部运营人员，内容安全守护者	1. 高效地审核用户上传的视频、评论、弹幕。 2. 对违规内容进行便捷处理（通过/驳回/封禁）。 3. 有明确的审核标准和操作指南
系统管理员	最高权限管理者	1. 管理所有用户和角色权限。 2. 配置系统全局参数。 3. 监控系统健康状况和数据报表

2.3. 关键涉众和用户的需要

2.3.1 视频消费者与互动者（如二次元爱好者）的需求

高质量内容获取：系统能够轻松发现并观看符合自身兴趣的高质量视频内容，满足娱乐、学习或深度文化消费的需求。

实时互动与归属感：在观看视频时，能通过弹幕、评论等方式与他人进行实时互动，感受“一起看”的陪伴感，并基于兴趣找到同好，建立强烈的社区归属感。

个性化体验：系统可以获得根据个人喜好定制的推荐内容，减少信息过载，提升内容发现的效率和愉悦感。

2.3.2 内容创作者（UP 主）的需要

便捷的创作与分享：系统拥有简单易用的工具来创作、上传和管理视频内容，并能轻松地将自己的见解或作品分享给目标受众，找到知音。

有效的反馈与成长：系统反馈清晰的播放量、粉丝增长、互动数据等反馈，以便让 up 主了解内容效果、优化创作方向，并建立个人影响力。

与粉丝深度互动：系统能够方便地让 up 主与粉丝进行交流（如回复评论、发布动态），维护和壮大自己的粉丝社群。

2.3.3 审核员的需要

高效与精准的审核：系统在面对海量内容时，能快速、准确地对视频、弹幕、评论进行合规性判断，显著提升审核效率，降低工作压力。

标准统一与决策支持：审核过程中有清晰、统一的规范和工具支持，确保审核尺度一致，减少误判和争议。

2.3.4 系统管理员的需要

系统稳定与可视化管理：系统能够全面监控平台运行状态（如服务器健康度、流量峰值），快速定位并处理问题，确保平台的高可用性和稳定性。

安全与可控的资源管理：系统具备强大的用户权限管理和数据资源控制能力，防止安全事件，保障平台和数据的安全有序。

2.4. 产品概述

2.4.1 产品定位陈述

for	**核心用户**：16-28 岁学生/年轻上班族（二次元、科技、知识领域爱好者） **细分群体**： 内容消费者（追求互动与归属感） UP 主（需要创作工具与流量支持） 审核员/系统管理员（保障内容安全与平台稳定）
who	**平台角色**： 兴趣社区连接者（弹幕+社群驱动） 创作生态赋能者（工具+数据支持） 内容安全守护者（审核+合规） **价值主张**： 对消费者：实时互动+个性化推荐 对创作者：公平流量+变现支持
the	**产品定位**： 以特定兴趣（如二次元、科技）为核心，集高质量视频、弹幕互动、深度社群于一体的垂直类视频社区平台，满足 Z 世代文化认同与深度互动需求。
That	**差异化优势**： 独特的弹幕文化（实时集体观看氛围） 强社区归属感（兴趣标签+社群） 创作者友好（简洁上传+数据反馈） 垂直领域聚焦（避免泛娱乐化竞争）

2.4.2 完整的产品概述

a 站是一款聚焦 Z 世代（16-28 岁）的垂直兴趣视频社区，以二次元、科技、知识等细分领域为核心，通过“高质量视频+实时弹幕+深度社群”模式，打造兼具娱乐性与文化归属感的兴趣聚集地。

目标用户涵盖内容消费者（追求互动与同好社群）、UP 主（需要创作工具与流量支持）及运营方（审核员、系统管理员）。消费者可通过智能推荐获取个性化内容，利用弹幕实时互动，并加入兴趣标签社群；UP 主可便捷上传视频、分析数据并获得创作激励；运营方则通过 AI+人工审核保障内容安全。

核心功能包括：

视频消费：多端观看、弹幕互动、智能推荐；

创作支持：一键上传、数据分析、收益分成；

社群运营：兴趣标签匹配、动态广场、私信互动；

管理保障：内容审核、青少年模式。

差异化优势在于：

弹幕文化：实时互动形成集体观看氛围，增强参与感；

社区属性：通过标签与社群构建深度连接，避免泛娱乐化；

创作者友好：简化流程、透明数据、公平流量，降低创作门槛；

垂直聚焦：深耕细分领域，精准满足用户需求。

发展愿景：短期积累 xxx 核心用户，签约 xxx 优质 UP 主；长期成为国内领先垂直兴趣社区，覆盖 xxx 用户，拓展电商、IP 衍生等增值服务，构建“内容-互动-变现”生态，实现可持续增长。

2.5. 产品特性 （需求**金字塔的底层**，是为了满足用户需要的功能）

对于内容消费者：可以提供智能推荐，比如经常点击哪个板块，还可以提供弹幕系统和聊天系统，增加内容消费者的参与程度，还可以添加点赞收藏评论功能，方便用户重复播放和分享个人观点或者吐槽，同时还可以关注 up 主，形成个人的动态视频流

对于 up 主（内容创作者）：可以提供视频上传工具，还可以分析视频的数据的工具进行反馈浏览，以及稿件的管理，封面与标题优化的建议，还有和粉丝互动的渠道，比如私信或者评论回复

对于审核员：应该有一个方便审核员审核的队列方便他驳回或者通过，并且可以添加驳回理由。或者可以的话实现一个内容详情的预览，方便审核是否违规或者插入自动检测屏蔽词自动驳回功能。

对于系统管理员：应该提供全局用户管理（封禁解封）以及系统的参数配置，云服务器情况，以及系统监控的仪表盘，数据的汇报情况导出或者展示功能

2.6. 其他产品需求

性能需求：

首页首屏加载时间 < 3 秒。

视频播放点击后开始缓冲时间 < 5 秒

安全性需求：

用户密码需进行处理后存储

敏感操作（如修改密码）需二次验证

可用性需求：

界面设计简洁直观，符合目标用户群体的审美

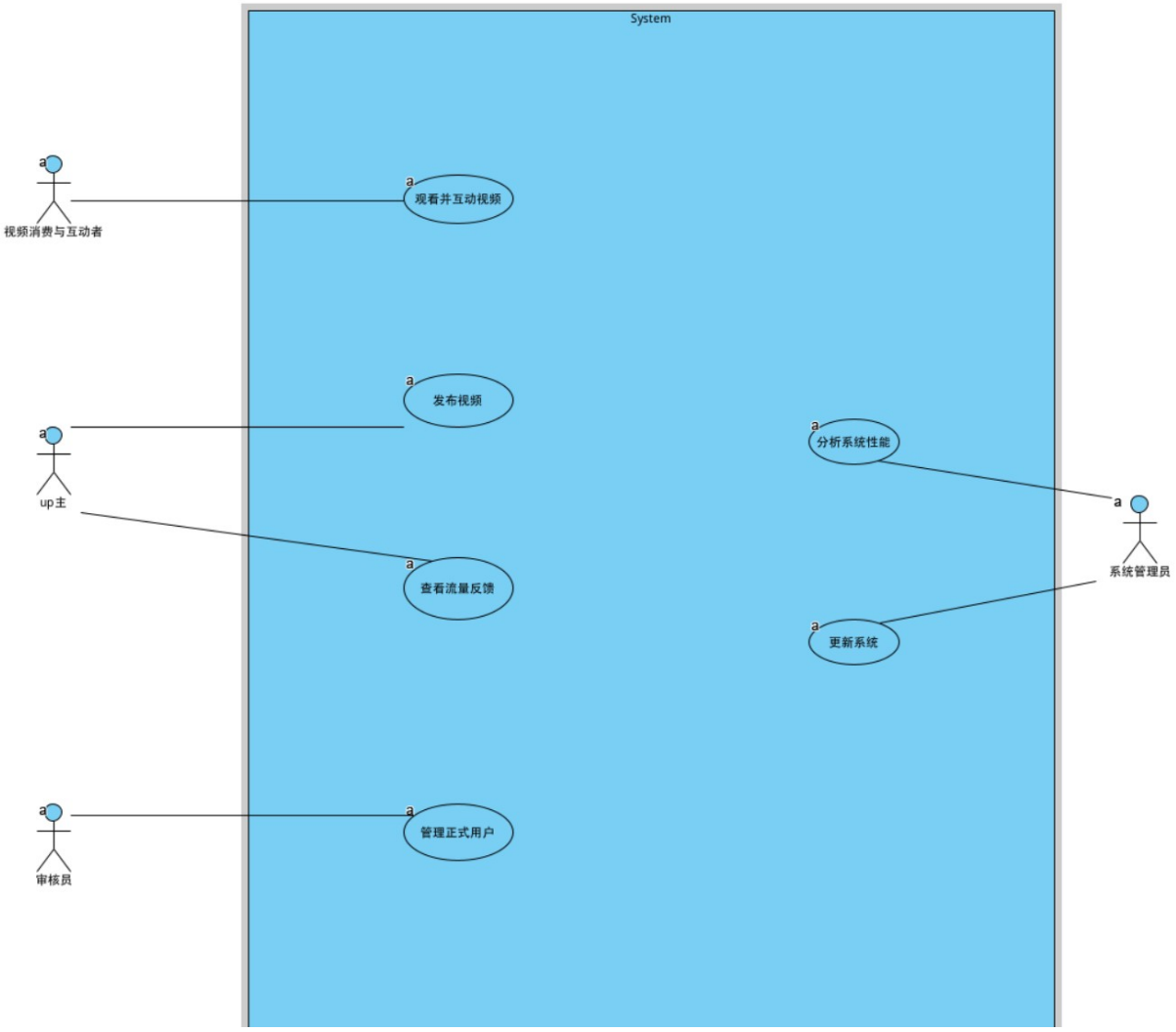
操作流程符合直觉，新用户无需培训即可完成核心操作（看视频、发弹幕）

3 用况建模

3.1. 术语表

名称	定义
UP 主	平台的内容创作者
个人消息	包含所有的回应信息，系统通知，聊天信息
弹幕	在视频播放时实时飘过的评论
一键三连	对视频同时进行点赞、投币、收藏的操作
动态	UP 主发布的非视频类图文内容，类似微博
AV 号/BV 号	视频的唯一标识符
粉丝	普通用户关注 up 主之后会自动变为其粉丝
内容消费与互动者	区别于 up 主，此参与者主要是为了看视频和与视频互动
正式用户	包括 up 主和内容消费与互动者的更大群体

3.2. a 站的主要用况



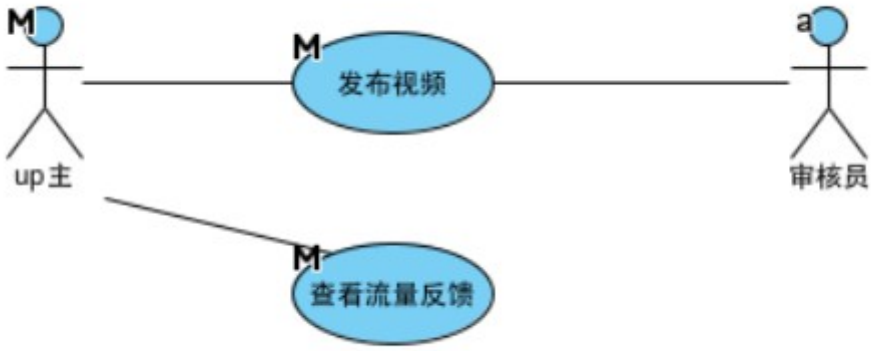
3.3. 各部分用况图以及简要的用况和参与者描述

3.3.1 第一部分：**视频消费与互动者**

视频消费与互动者：视频消费与互动者是负责观看视频和与视频进行互动的人。

观看并互动视频：系统向视频消费与互动者提供了观看视频和与视频互动的功能，与视频互动包括了很多小功能，比如点赞，投币，收藏分享，关注和评论以及回复的功能。

3.3.2 第二部分：up 主



up 主（内容创作者）：**up 主**是为了凸显 **up 主**和区别**视频消费与互动者**的参与者，他主要是分享视频到系统。

发布视频：系统向 **up 主**提供了可以上传视频的功能，系统验证视频格式并处理上传，视频进行审核成功后将推送到主界面。

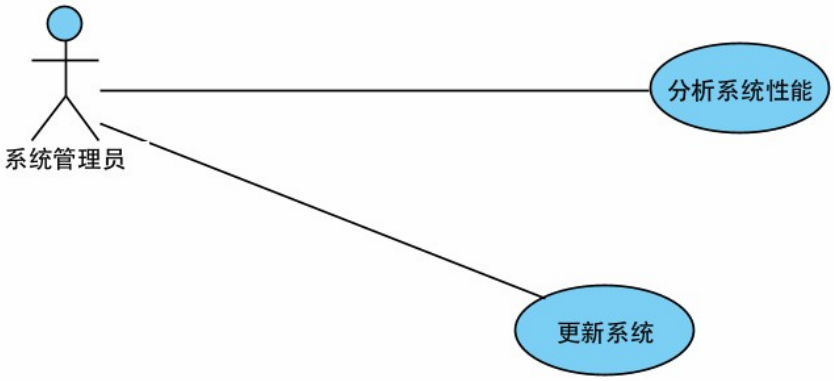
查看流量反馈：系统向**up 主**提供了一系列的流量反馈机制，比如查看视频被投币数量，被点赞数，视频播放量等，还可以根据各项数据反馈激励资金等。

3.3.3 第三部分：**审核员**

审核员：审核员是负责审核正式用户所发出的各种各样的内容（包括弹幕评论，视频内容，视频，动态）的人，拥有着驳回和同意视频或动态上传到首页或者动态朋友圈，删除评论或者弹幕的权力。

管理正式用户：管理正式用户的视频本身和视频的描述，决定通过或者驳回视频发布;管理视频的弹幕和评论，移出违规内容;管理正式用户的账户，处理违规行为，决定封禁或者解除封禁。

3.3.4 第五部分：**系统管理员**



系统管理员：是负责整个系统的最高权限人，可以拥有配置系统全局参数的权限。

更新系统：系统管理员执行预定的更新流程，将更新的内容进行封装更新并发布。

分析系统性能：系统管理员分析系统性能，通过登录监控平台进行日常巡检，审视用户体验、应用

服务、数据及基础设施等各层级的核心指标。

3.4. 用况的详细描述

3.4.1 第一部分：视频消费与互动者用况

3.4.1.1 观看并互动视频

观看并互动视频基本流：

参与者：视频消费与互动者

前置条件：1.参与者视频消费与互动者的用户设备与互联网连接正常。

2.参与者视频消费与互动者参与互动需要已登录并通过身份验证。

3.视频资源在系统中必须存在且可访问。

后置条件：1.视频的观看次数已更新。

2.参与者视频消费与互动者的互动数据（比如点赞，评论等）已持久化存储，并关联到对应视频。

3.参与者视频消费与互动者的观看历史已经刷新，包含本次观看视频并互动的详细信息（比如视频的id,视频的观看时长，观看时间等信息）。

4.系统日志中已记录用况的执行摘要（包括任何错误代码）。

基本流：

1.当参与者**视频消费互动者**进入系统后，用况启动。

{搜索视频}

{登录系统}

2.系统将会加载首页的推荐页下的视频列表，视频列表的视频将会显示视频的封面和名字，up主头像，名字和播放量。

3.参与者点击推荐页下的视频列表。

4.系统将加载视频页的基本信息：

·左上角的回到主页按钮，右上角的最小化，最大化，退出按钮。

·左侧的视频区域，右侧的互动区域

·左侧的视频下方区域将显示播放/暂停，当前时长/总时长以及对应的视频进度条，弹幕开关按钮和弹幕设置按钮和发送弹幕的对话框，分辨率设置，视频播放倍数，视频页设置和音量以及全屏按钮。

·右侧互动区域分成两个板块：简介和评论区。

简介区将显示 up 主头像以及名称和粉丝数，以及关注按钮，下方将会显示视频播放，评论数量以及视频发布时间和可以点击展开按钮展开视频简介。下方将显示点赞，投币，收藏，缓存，分享等按钮以及下面会推荐视频。

评论区将显示输入评论的评论框和发送按钮，以及每个人的评论列表和对评论的回复列表，评论和回复列表都会显示对应的头像，名字，评论时间，评论内容，点赞数，点踩数。

5.系统加载完毕后会自动播放视频并记录视频播放量+1。

{视频控制}

6.系统将会记录参与者看的对应视频。

7.参与者点击视频右侧互动区域进行互动操作：

前置条件：参与者必须进行登陆之后才可以进行互动行为

7.1 参与者进行点赞操作：系统将会记录是哪个参与者给对应视频点赞，并显示点赞数量+1,点赞图标显示变红，如果再次点击将会取消点赞，系统记录取消点赞行为，点赞数量-1,点赞图标变黑。

7.2 参与者进行投币操作：系统将会记录是哪个参与者给对应视频投币，并显示投币数量+1,可以多次

投币，系统首先会检测参与者的硬币数量是不是大于 1,系统记录每一次的投币信息，不可取消投币。并且每次投币会将参与者的硬币数-1.

7.3 参与者进行收藏操作：系统将会记录是哪个参与者收藏了对应视频，并显示收藏数+1，收藏图标变成完整的⭐，系统记录收藏数+1.

7.4 参与者进行缓存操作：系统将会记录是哪个参与者缓存了对应视频，并显示已经缓存，缓存将会保存到本地，同时系统记录已经缓存，再次点击将会无效不能再次保存。

7.5 参与者进行评论操作：参与者点击评论列表输入内容发送后，系统将会把参与者评论的内容显示到评论列表当中，系统将会记录评论内容，评论时间。

7.6 参与者进行回复操作：参与者点击评论回复后，系统将会显示出回复对话框，参与者输入回复内容之后点击发送，系统将会把内容显示到评论的回复列表当中并保存。

8.参与者点击右上角的 x 之后，系统将会回到主页推荐界面，用况结束。

备选流：

A1 “搜索视频”（普适备选流）

触发条件：任何时候视频消费与互动者在浏览首页推荐视频或正在观看视频的时候，点击了首页上方的搜索框。

1.参与者输入关键词到搜索框。

2.系统根据关键词搜索对应视频，然后将搜索到的视频展示到搜索结果页面，如果找不到对应视频，

将显示没有找到相关视频的提示。

3.参与者选择：

3.1 参与者点击了搜索界面视频列表的视频，如果有视频，系统将退出当前视频然后事件流恢复到基本流第 4 条。

3.2 参与者重新输入关键字搜索视频，事件流恢复到备选流 A1 的第 2 条。

3.3 参与者点击右上方的退出按钮，用况结束。

A2“登录系统”（普适备选流）

触发条件：任何时候视频消费与互动者在浏览首页推荐视频或正在观看视频的时候，点击了首页左边的登录按钮。

1.参与者点击首页左边的登录

2.系统将弹出登录界面，登录界面将会显示账号，密码输入框，记住密码勾选框，登录，立即注册按钮。

3.参与者选择如下情况：

3.1 参与者输入账号密码点击登录之后，系统检测没有该账号或者有该账号，但是密码错误：系统将显示登录失败，账号或密码错误。

3.2 参与者输入账号密码点击登录之后，登录检测有该账号，并且密码正确：系统将该账号数据给当前设备。

3.3.1 参与者点击立即注册：系统将弹出注册界面，注册界面显示填写账号，密码，确认密码，昵称等输入框以及我已阅读并同意《用户协议》和《隐私政策》。

3.3.2 参与者输入对应的账号密码等内容，如果有重复账号系统将会提示注册失败，账号可能已存在或者密码和确认密码有误，系统会弹出两次输入的密码不一致。

3.3.3 参与者注册后，系统会回到登录界面并保存参与者的账号密码昵称：信息。

3.4 参与者取消登录：参与者点击其他空白地方或者登录界面或者注册界面的 x,将退出登录或者注册界面。

事件流恢复：登录成功或取消登录注册后事件流将回到触发登录系统时的界面。

A3“视频控制”(有界备选流)

触发条件：在视频播放的过程中

1.参与者滑动鼠标到视频区域。

2.系统显示各项控制视频的控制选项：左侧的视频下方区域将显示播放/暂停，当前时长/总时长以及对应的视频进度条，分辨率设置，视频播放倍数，音量以及全屏按钮。

2.1 参与者点击播放/暂停按钮：系统将会暂停或者继续播放视频，事件流恢复到 A3 备选流的第 2 条

2.2 参与者点击进度条：系统将会跳转到视频的对应位置，并显示对应进度的视频内容。事件流恢复到 A3 备选流的第 2 条

2.3 参与者点击分辨率设置：可以选择 720p 或者 1080p,系统将会根据参与者选择的分辨率调整视频清晰度并刷新视频然后显示给参与者。事件流恢复到 A3 备选流的第 2 条

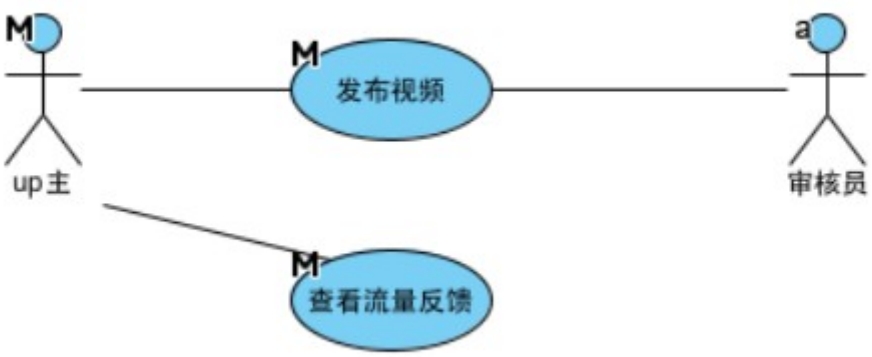
2.4 参与者点击倍数：系统将显示倍数 0.5,0.75 到 2.0 依次 0.25 提升的倍速，参与者选择后，系统将根据参与者选择的倍数调整视频播放速率并显示到播放区域。事件流恢复到 A3 备选流的第 2 条

2,5 参与者点击音量：系统将显示竖着的音量条，可以调整音量，参与者调整音量后，系统将根据参与者给定的音量调整视频声音。事件流恢复到 A3 备选流的第 2 条

2.6 参与者点击视频界面右上角的 x,备选流结束，系统将恢复到当前系统界面的状态。

4 需求分析

4.1. 健壮性分析



4.1.1 UP 主

对“发布视频”用况进行健壮性分析：

发布视频包括：up 主上传视频，设置封面、文案等并发布。

首先确定边界类：发布视频用况涉及的边界有视频发送界面（VideoSendUI)。

确定控制类：处理视频发送界面的控制器为（VideoSendController）。

确定实体类：对于 VideoSendController,涉及的实体类为 Video 视频类，用于存储视频相关信息（例如视频时长等）。

对“查看流量反馈”用况进行健壮性分析：

首先确定边界类：管理视频用况涉及的边界有视频简介界面（VideoDescriptionUI)

确定控制类：处理视频简介界面的控制器为（VideoDesController）,处理视频评论界面的控制（CommentController）。

确定实体类：对于 VideoDesController,涉及的实体类为 Video 视频类，用于获取视频标题、视频封 面、视频文案等。对于 CommentController，涉及的实体类为 Video 视频类和 Comment 评论类，Video 类用于获取当前视频，Comment 类用于获取所有评论。

用况简要描述：

对 “查看视频数据（流量机制）” 用况进行健壮性分析：

查看视频数据（流量机制）用况包含： up 主通过创作者中心进入数据分析模块，查看已发布视频的各项流量与互动数据，包括播放量、点赞、收藏、评论等，支持时间范围筛选与图表展示。

首先确定边界类： 创作者中心界面（CreatorCenterUI）一个边界类。

确定控制类： 处理数据分析请求的控制器为（DataAnalysisController）。

确定实体类： 存储视频基本信息的实体为（Video），存储用户流量统计数据的实体为（UserTrafficData）。

4.1.2 视频消费与互动者

对观看并互动视频用况进行健壮性分析：

观看并互动视频包括：参与者可以观看视频，并对视频进行分辨率、音量等设置;在简介页面，参与者可以查看 up 主信息、视频信息，还能够对视频进行点赞、收藏等操作;在视频的评论区域，可以发布评论、回复其他用户，并对评论和回复点赞。

首先确定边界类：观看视频并互动用况涉及的边界有视频内容界面（VideoPlayerUI)、视频简介区域(VideoDescriptionUI)、视频评论区域(CommentUI)三个边界类。

确定控制类：处理视频内容区域的控制器为（VideoPlayController），处理视频简介区域为（VideoDesController），处理评论区域为（CommentController）。

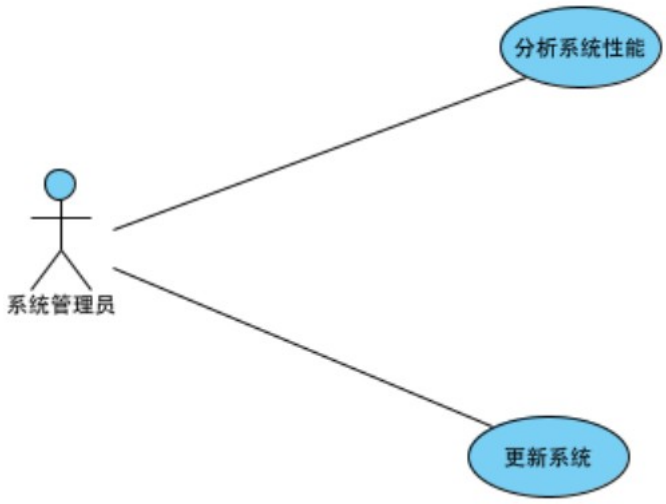
确定实体类：对于 VideoPlayController,涉及的实体类为 Video 视频类，用于获取视频内容、视频时长信息。对于 VideoDesController,涉及的实体类为 Video 视频类、Verified User 正式用户类，Video 用于获取视频标题、视频播放量、视频点赞量等。Verified User 用于获取视频上传者名字、粉丝数、头像。对 CommentController,设计的实体类为 Video 视频类、Verified User 正式用户类、Comment 评论类。Video 获取对于视频的唯一 id,Verified User 正式用户类 于获取评论或者回复的对应用户，Comment 用于获取评论和回复的数据。

4.1.3 审核员

对“管理正式用户”用况进行健壮性分析：

管理正式用户：管理正式用户的视频本身和视频的描述，决定通过或者驳回视频发布;管理视频的弹幕和评论，移出违规内容;管理正式用户的账户，处理违规行为，决定封禁或者解除封禁。
首先确定边界类：审核员在管理正式用户界面审核，所以有一个边界类（ManageUserUI）。
确定控制类：审核员管理正式用户有一个控制器类（ManageUserControl）。
确定实体类：因为要审核视频，评论，用户，所以有三个实体类（Video,Comment,User）

4.1.4 系统管理员



对“分析系统性能”用况进行健壮性分析：

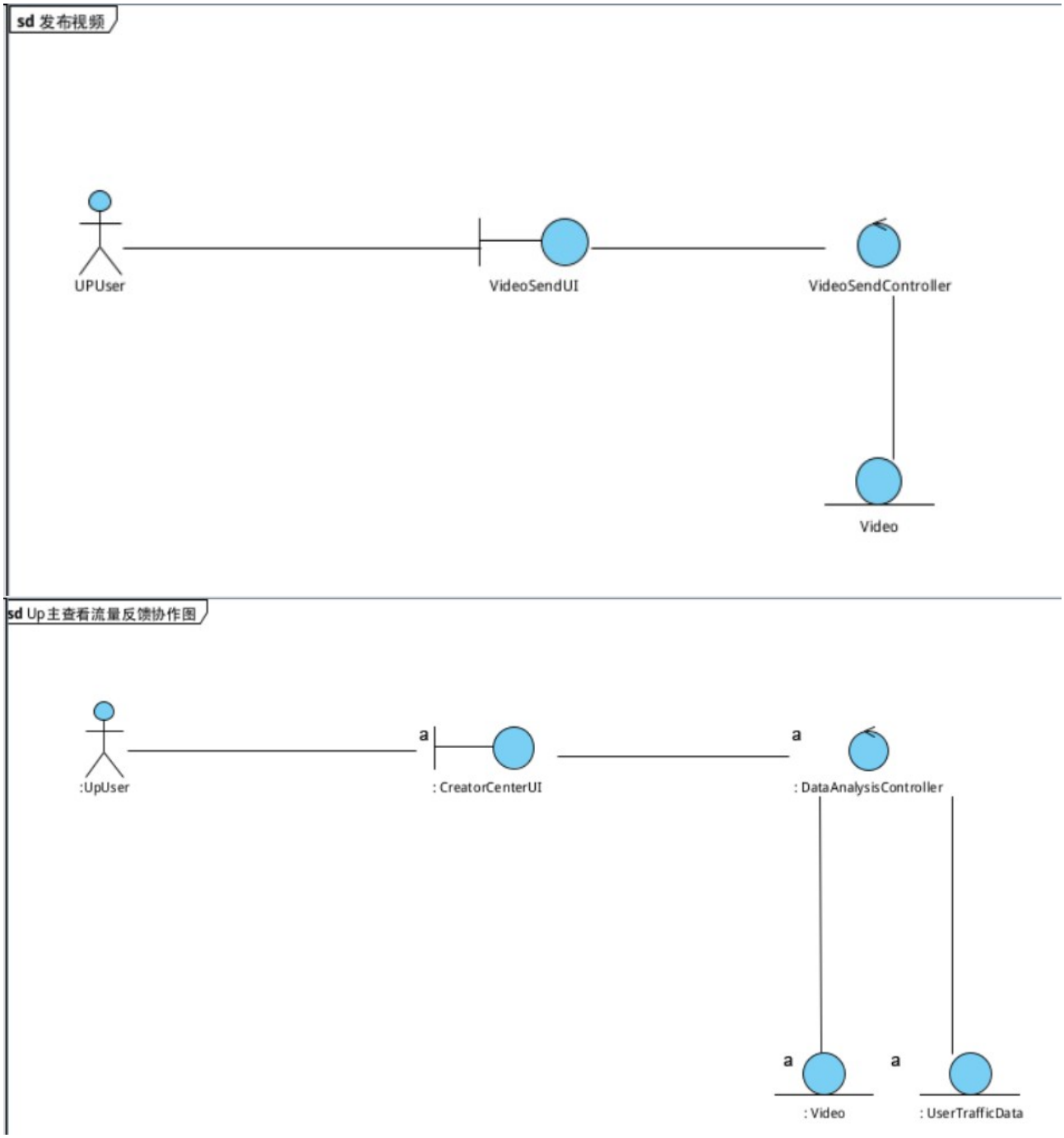
分析系统性能：登陆确认管理员身份后进行日常巡检， 检查数据和警告等，当监控系统发出告警时，立即响应，进行分析和处理，针对异常数据指标进行下钻分，深入查看关联指标定位问题范围
确定边界类：loginPageUI（登陆页面，用于确定管理员身份）DashboardUI（仪表盘数据页面），AlertManagermentUI（告警管理页面）
确定控制类：PerformanceAnalysisController（性能分析控制：发现问题）AlertManagementController（告警管理控制器）， ResourceManagementController（资源管理控制器）
确定实体类：serveSource（服务型资源实体），performanceMetric（性能指标实体），alert（告警实体）

4.2. 交互建模

4.2.1 UP 主

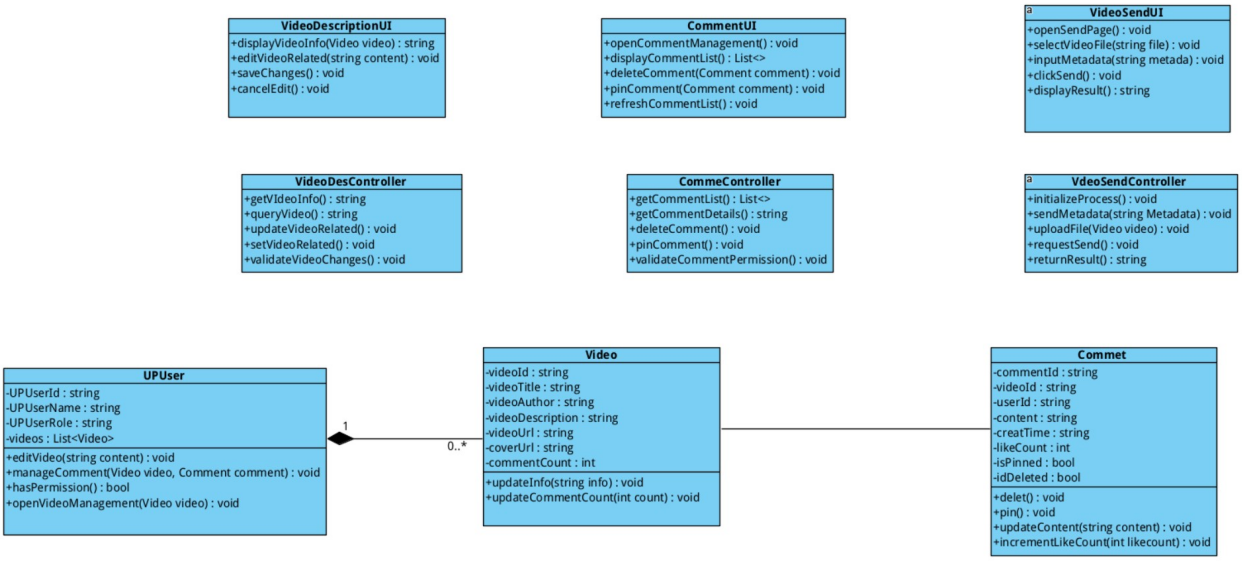
4.2.1.1 协作图

根据 4.1.1 中对 UP 主相关用况进行健壮性分析，已经初步确定了主要参与的实体对象，与其连接的边界类和控制类，见下图。

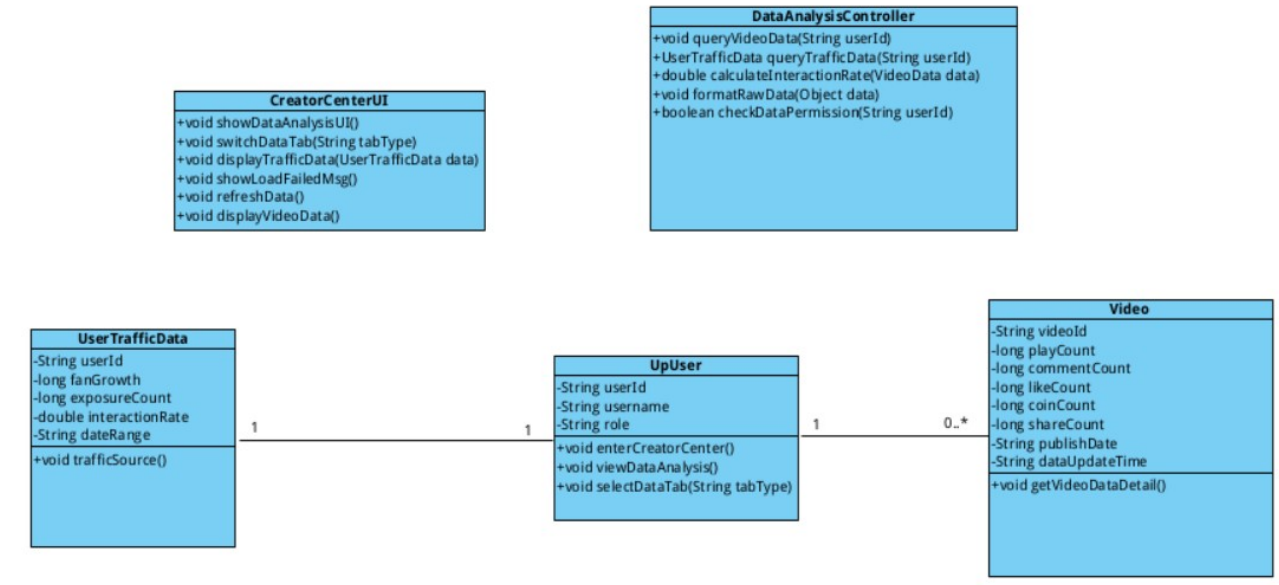

sd 发布视频
UPUser
VideoSendUI
VideoSendController
Video
sd Up主查看流量反馈协作图
:UpUser
a
: CreatorCenterUI
a
: DataAnalysisController
a
: Video
a
: UserTrafficData

4.2.1.2 类图

up 主用况“发布视频”的分析类图，见下图。



up 主用况“查看流量反馈”的分析类图，见下图。

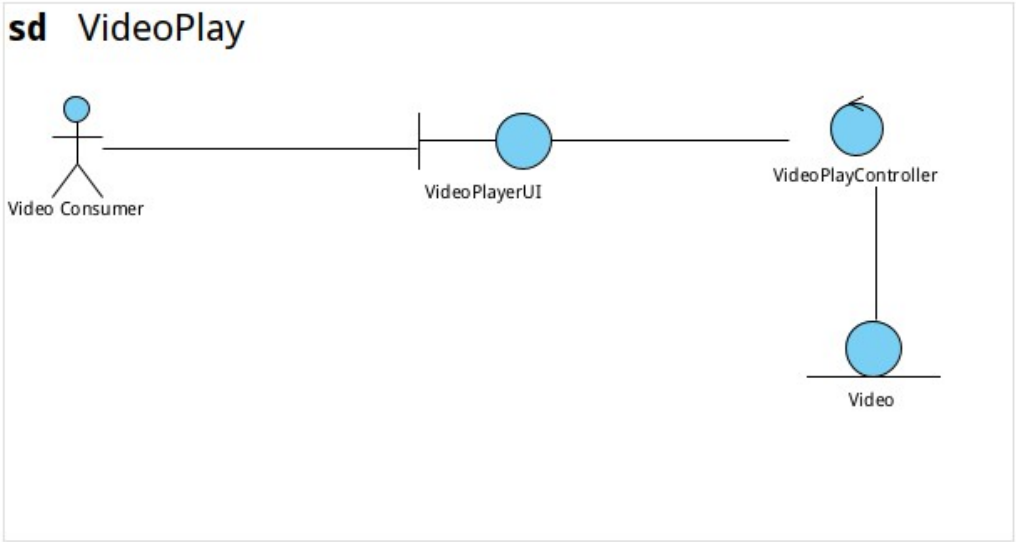


4.2.2 视频消费与互动者

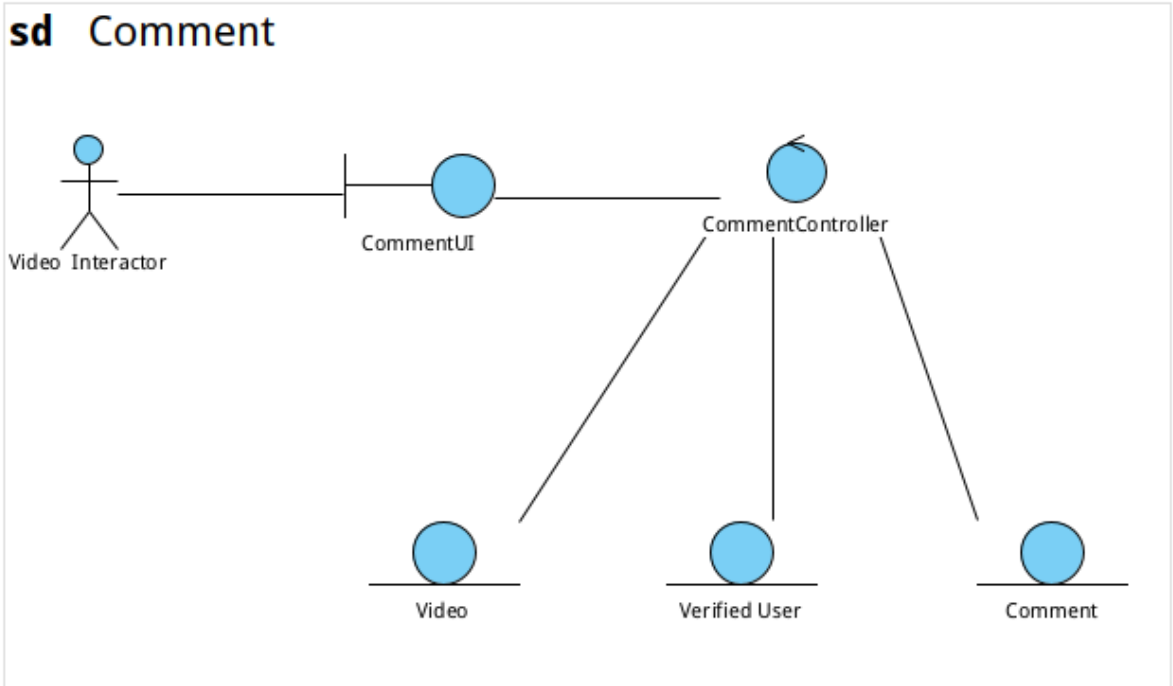
4.2.2.1 协作图

根据4.1.3中对正式用户相关用况进行健壮性分析，已经初步确定了主要参与的实体对象，与其连接的边界类和控制类，见下图。

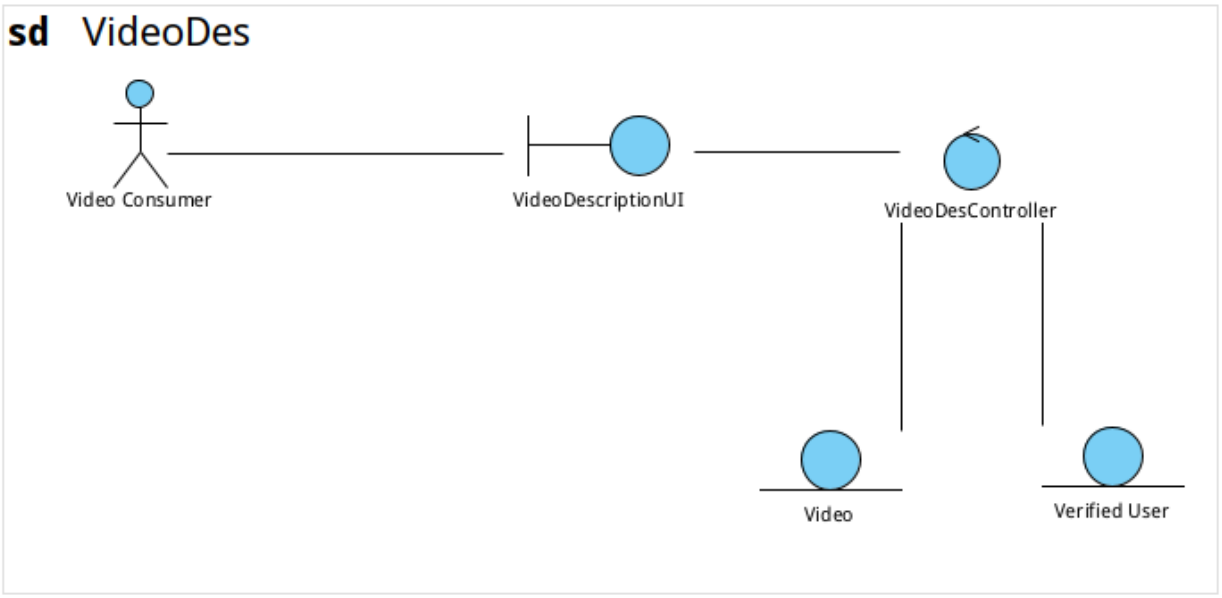
观看视频并互动播放视频情景：



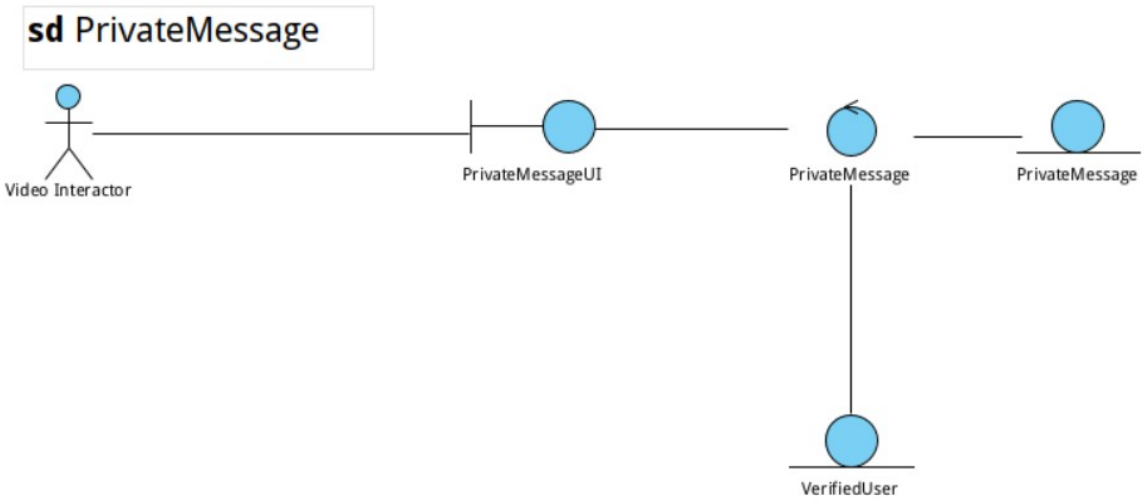
观看视频并互动评论情景：



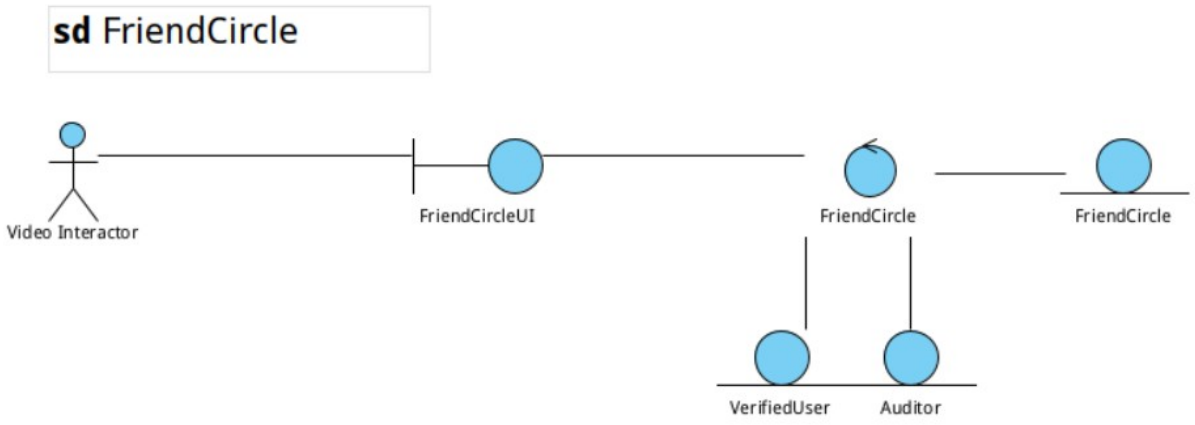
观看视频并互动视频简介情景：



与其他正式用户交流私信情景：



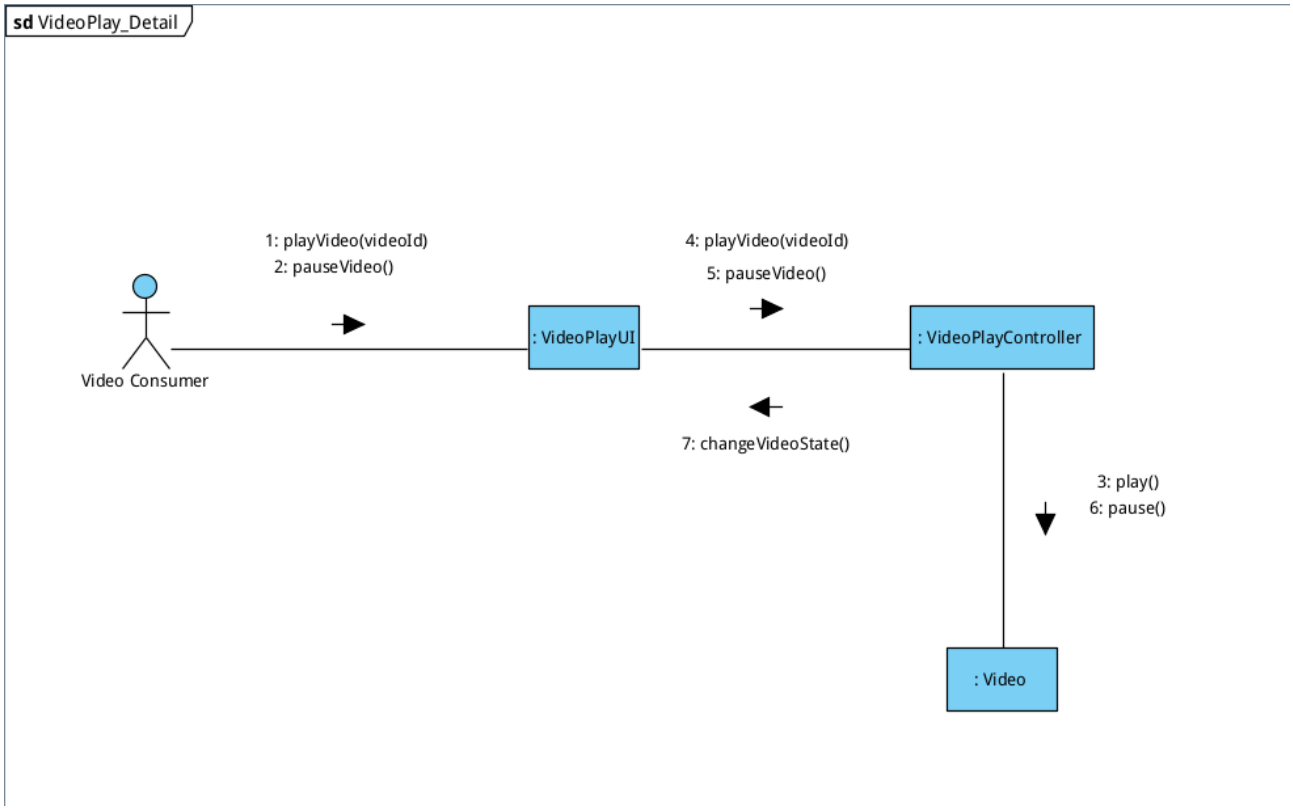
与其他正式用户交流朋友圈情景：



4.2.2.2 通讯图

现在来绘制协助的通讯图：加入通讯图的边框，及在各个实体类、边界类、控制类之间的消息与连接，见下图。

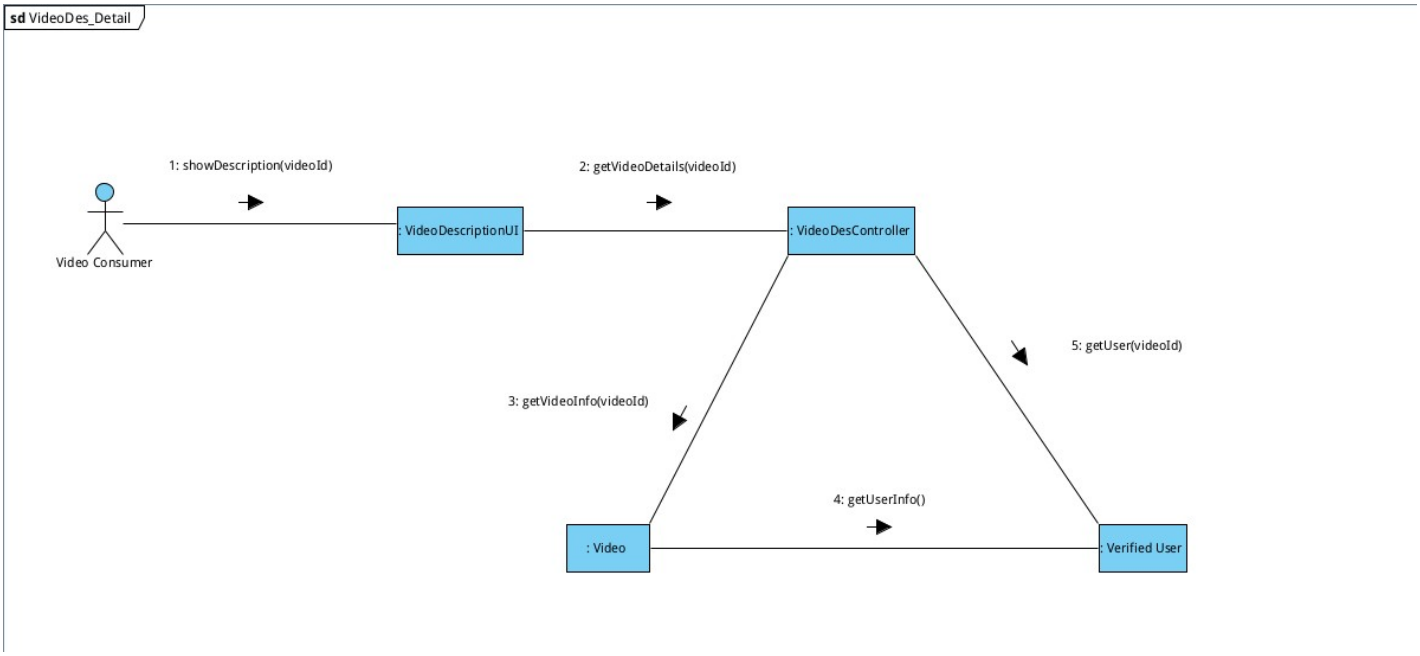
视频播放通信图：



视频用户通过前端 ui 界面点击播放视频、暂停视频，ui 将这个消息通知给控制器对象，控制器发送

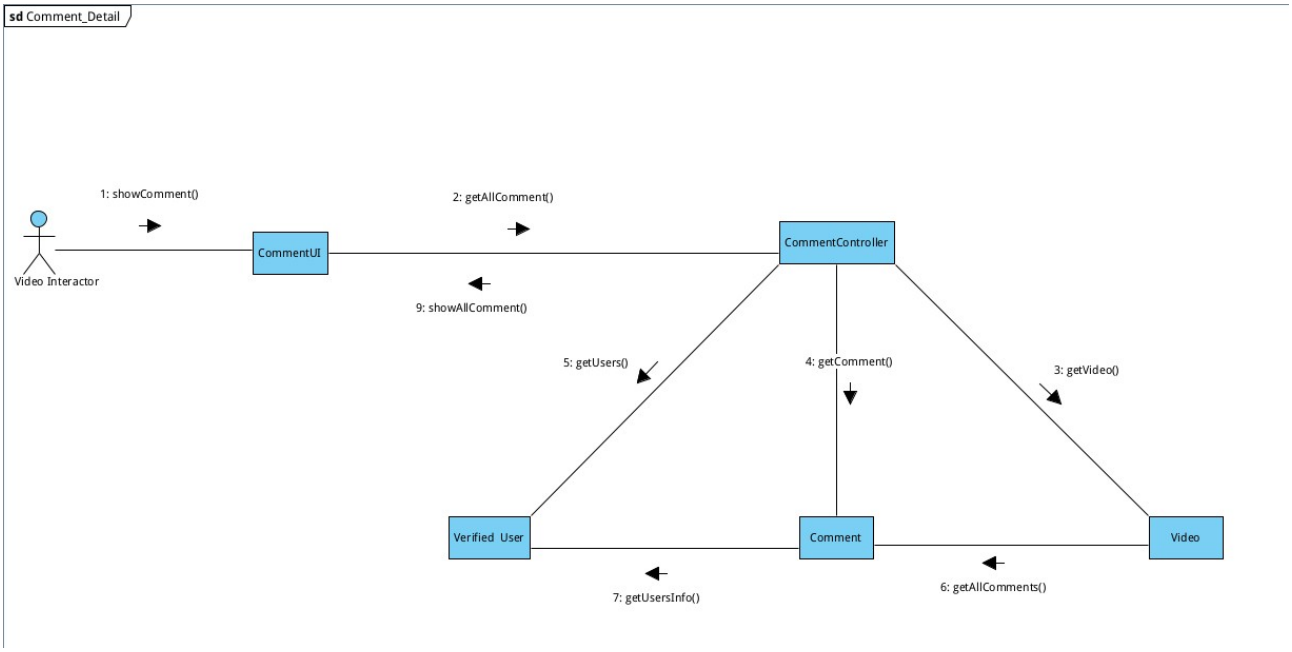
消息给视频对象去执行具体的业务逻辑。

视频简介通信图：



视频用户当打开播放界面的时候，会显示视频内容以及视频的介绍和对应的作者，因此需要控制器去连接视频类和用户类，将信息合并后显示在 ui 上。

视频评论通信图：

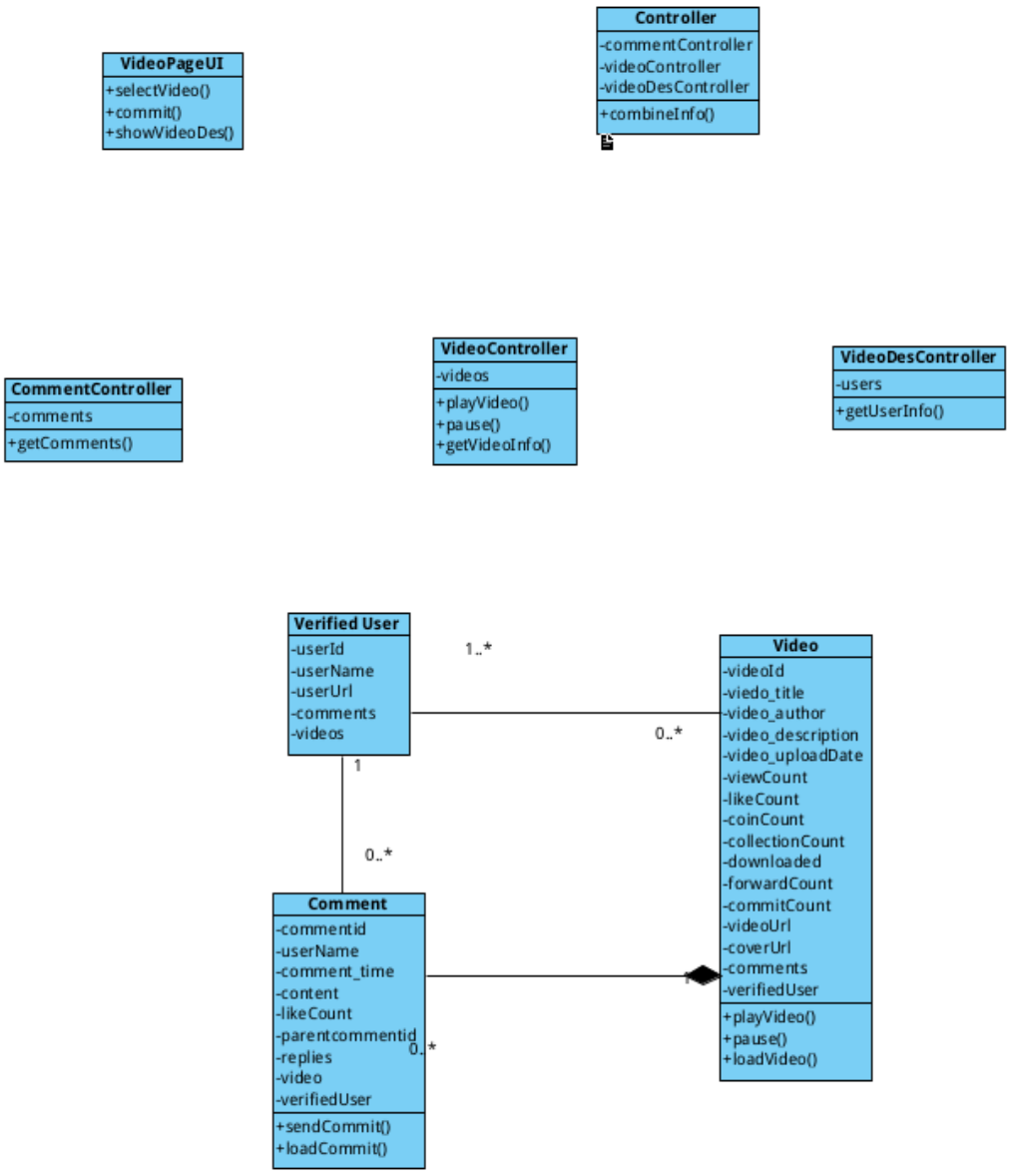


当用户打开评论区的时候，控制器去获取对应视频的用户的信息、评论的消息，最后将三个消息整

合显示给用户。

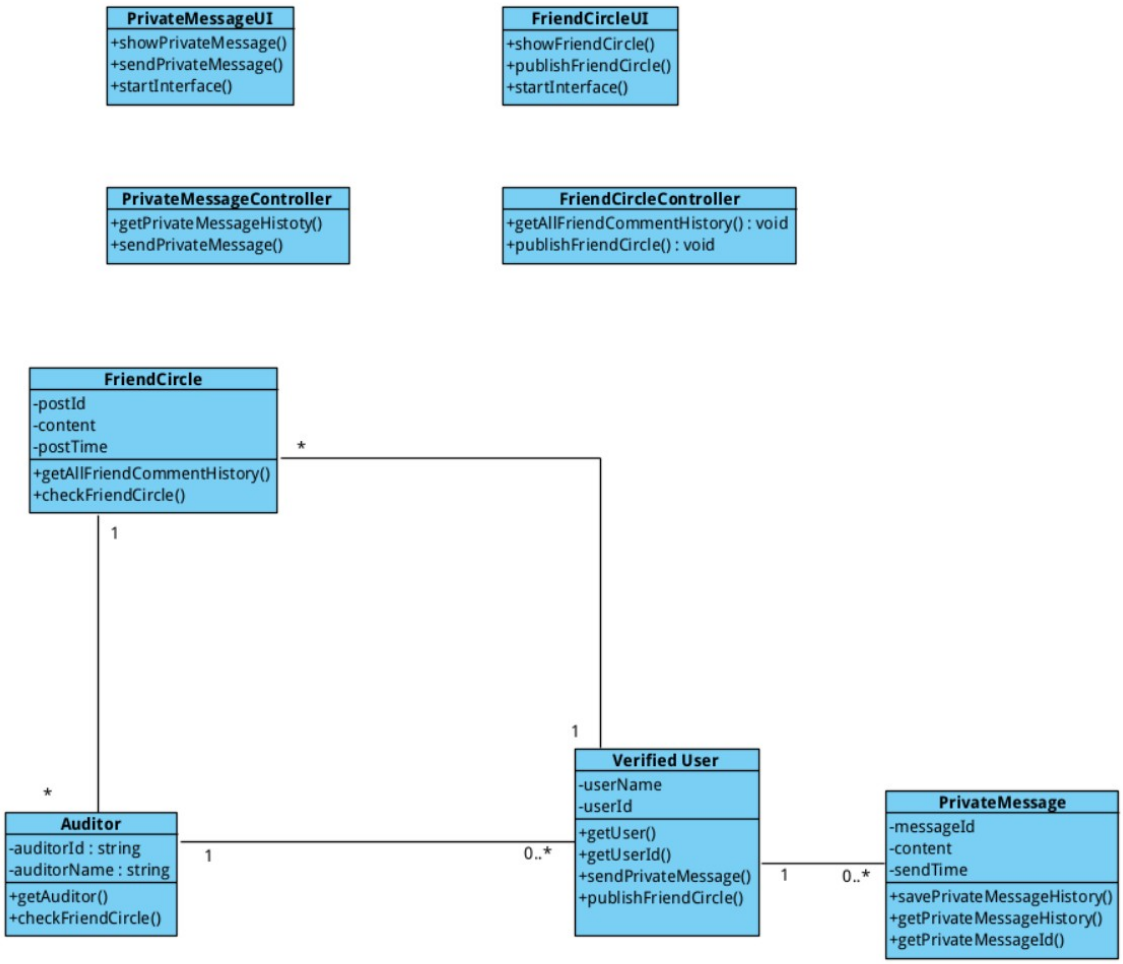
4.2.2.3 类图

内容消费与互动者用况“观看视频并互动”的分析类图，见下图。



总控制器 Controller 中依赖三个控制器 CommentController、VideoController、VideoDesController,这三个控制器分别去控制器对应的类;视频类中拥有 comment,是组合关系，视频与评论和正式用户有关联，一个用户可能有 0 个或者多个视频，一个用户可能有 0 个或者多个评论。

内容消费与互动者用况“与其他正式用户交流”的分析类图，见下图。

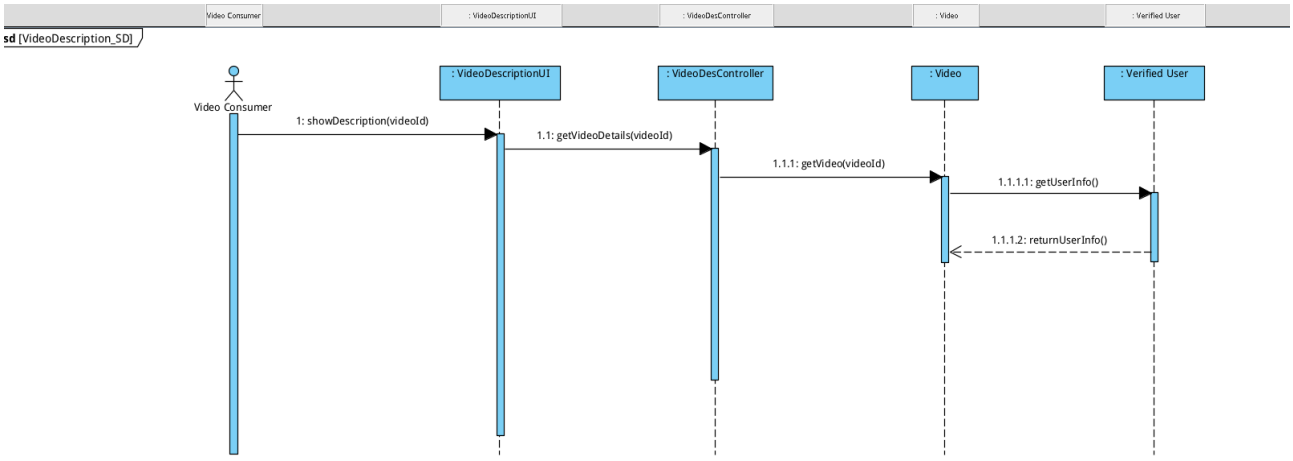


控制器 PrivateMessageController 和 FrinedCircleController 分别控制对应的 FriendCircle 和 PrivateMessage 实体类，UI 类有开始交互 startinterface（ ）开始显示以及对应展示内容的操作，实体类存储信息和获取，保存内容。verifiedUser 和 Auditor,一个是正式用户负责发送私信和发布朋友圈，审核员负责审核朋友圈，朋友圈实体类对应发送信息告知对应审核员，私信类则是负责保存历史记录和获取历史记录。一个 verifiedUser 正式用户可以对应多个朋友圈和多个私信，审核员可以审核多个正式用户，朋友圈通知多个审核员审核。

4.2.2.4 顺序图

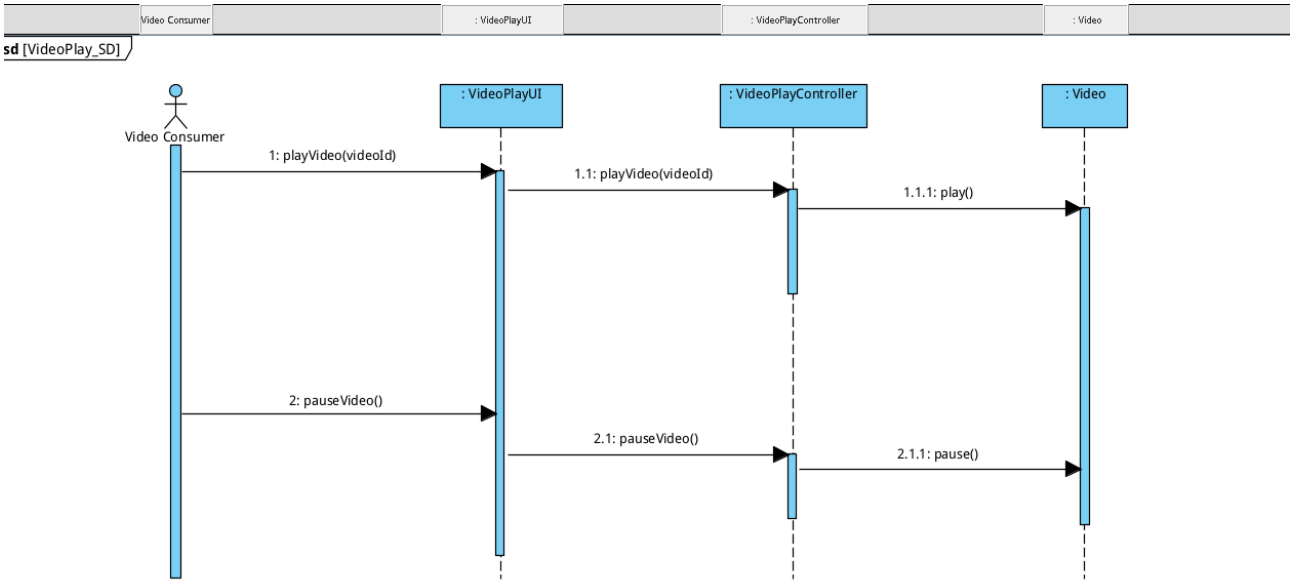
根据前述内容，我们绘制了关于用况“观看视频并互动”的顺序图，见下图。

视频简介顺序图：



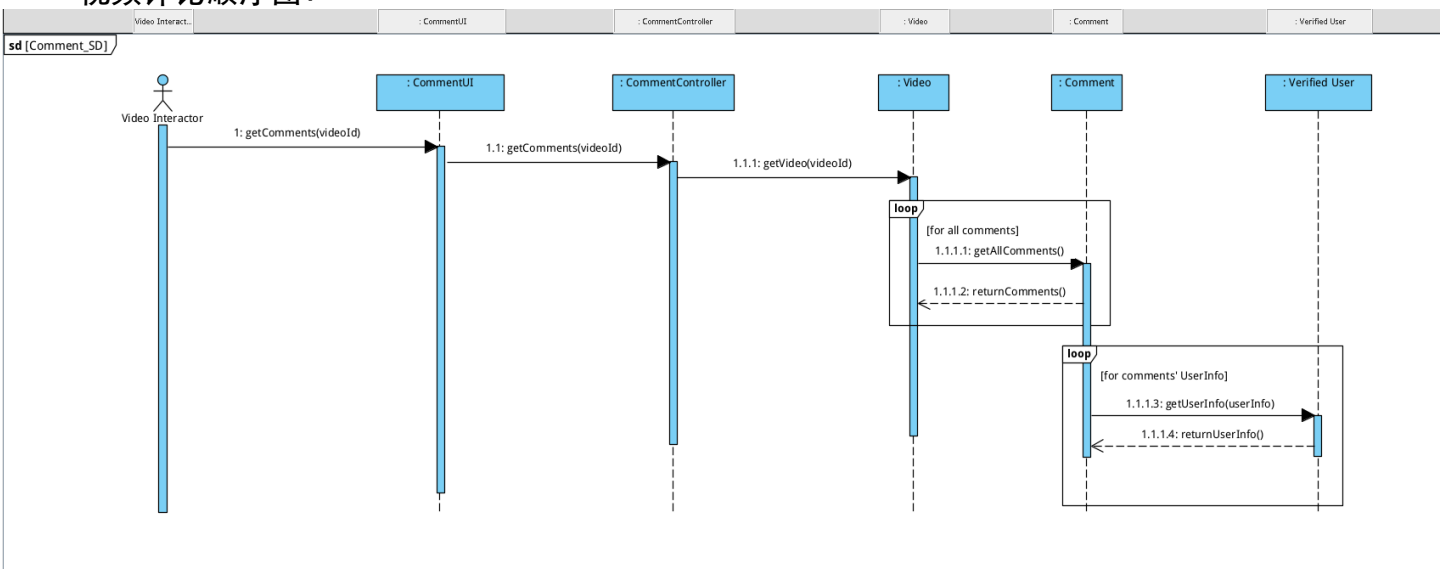
视频用户想要看简介的时候，ui 通知控制器，控制器通知对应的视频，获取视频信息，同时获取对应作者的信息，最后整合显示在 ui 上。

视频播放顺序图：



视频用户想要观看视频时，ui 通知控制器，控制器通知视频对象获取视频的信息，然后播放视频，显示在 ui 上，暂停视频的操作相同。

视频评论顺序图：

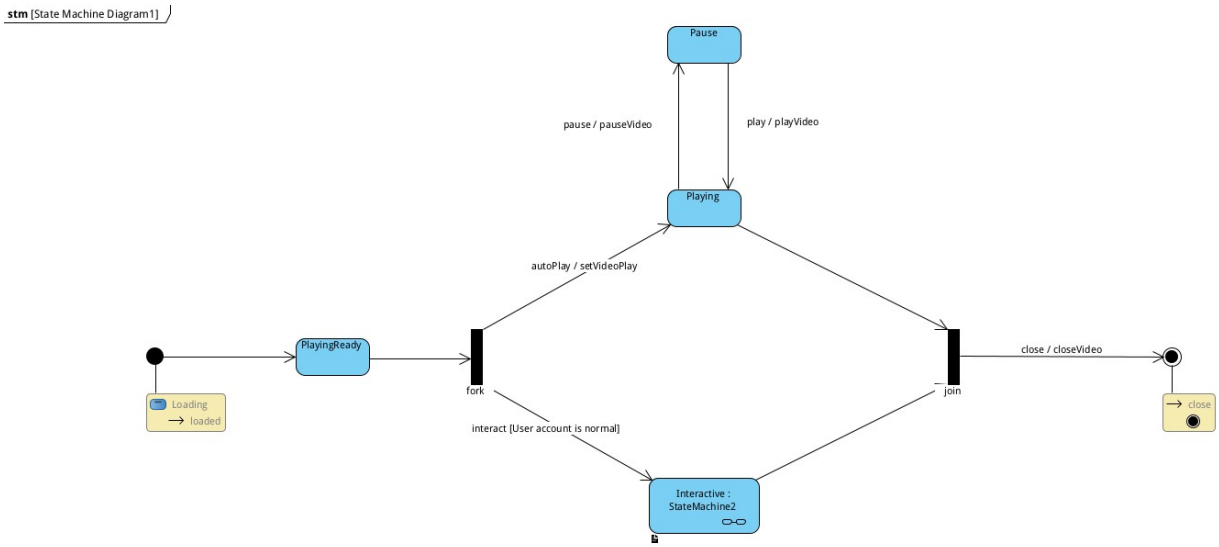


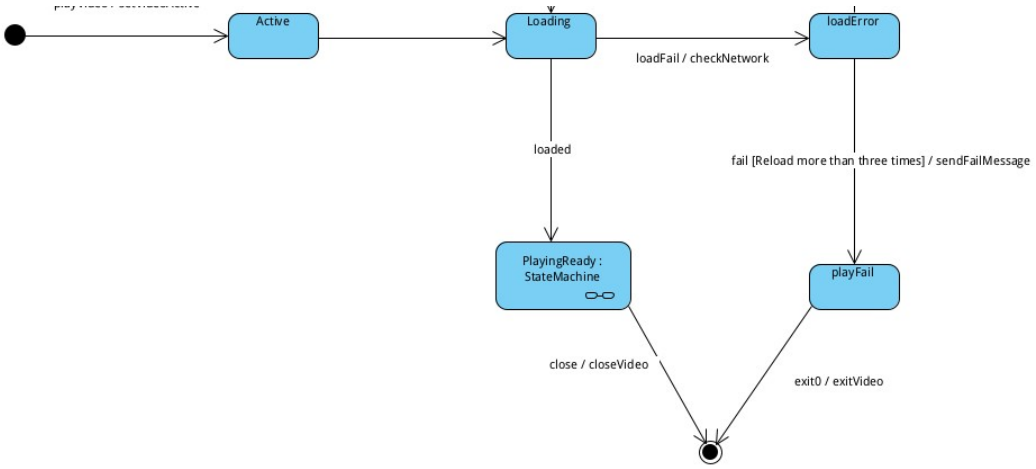
当视频用户查看评论区的时候，评论 ui 通知评论控制器首先获取视频的信息，视频去获取评论数据，每一条评论都会获取对应的基础用户信息，最后显示在 ui 上。

4.2.2.5 状态机

下面我们用状态机来表示单个 Video 对象对于它所有参与的用况的所有响应，见下图。

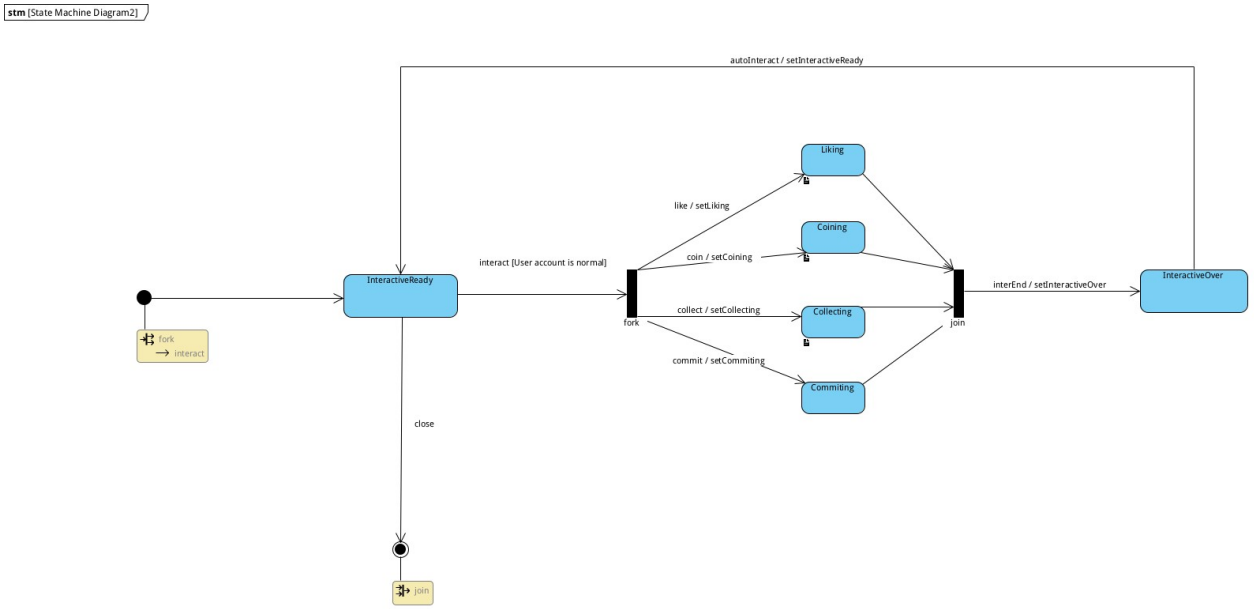
观看视频并互动的状态机：





进入播放准备状态后，可以同时观看视频和评论区，观看视频还可以暂停和继续，当两个操作都结束后就回到关闭视频状态，结束;如果想要评论交互，进入交互状态。

Interactive 层：

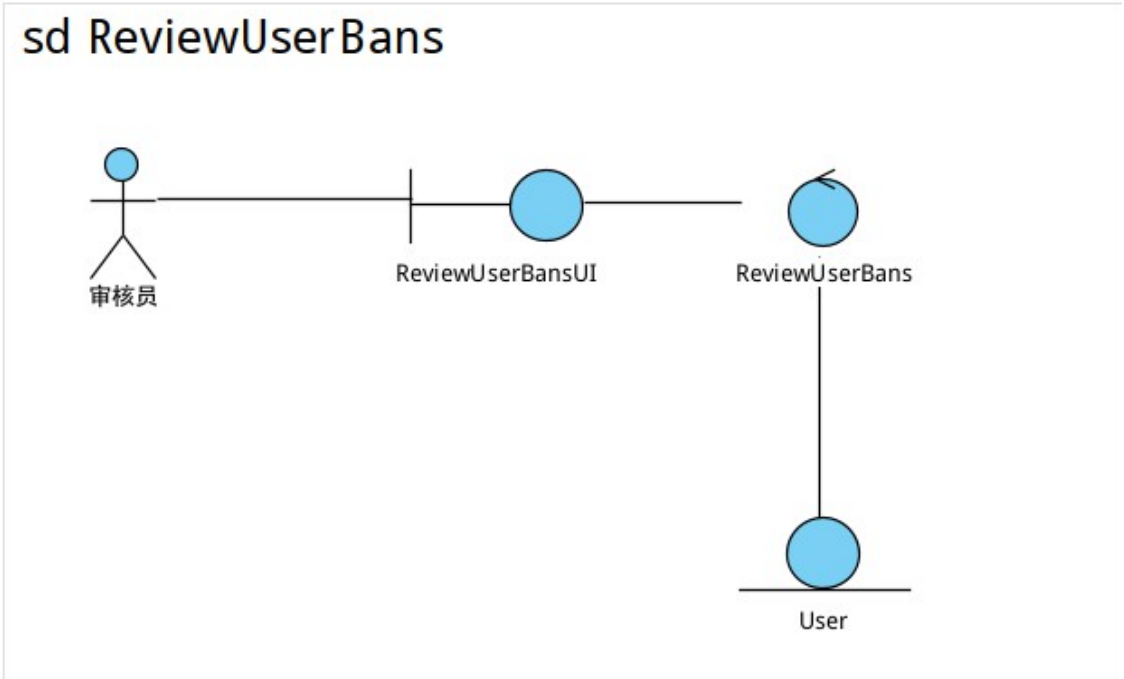


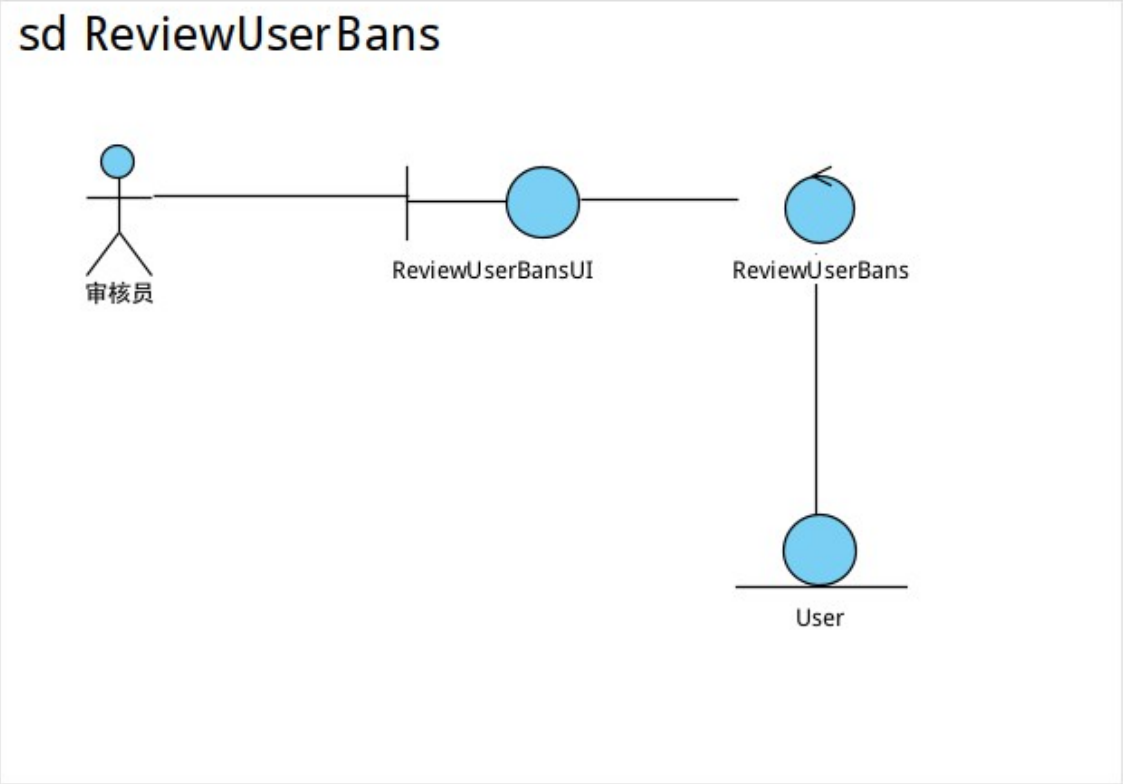
在交互状态中，可以同时执行点赞、投币、收藏、评论的操作，且都可以重复操作，当退出评论后结束交互状态。

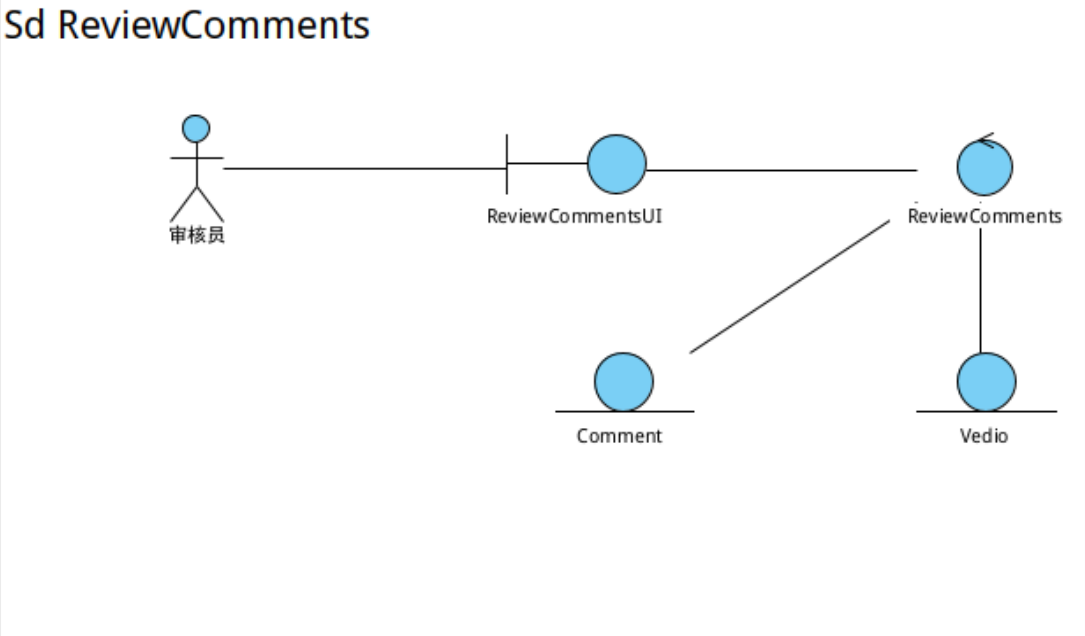
4.2.3 审核员

4.2.3.1 协作图

根据审核员的健壮性分析可以画出审核员管理正式用户的协作图。

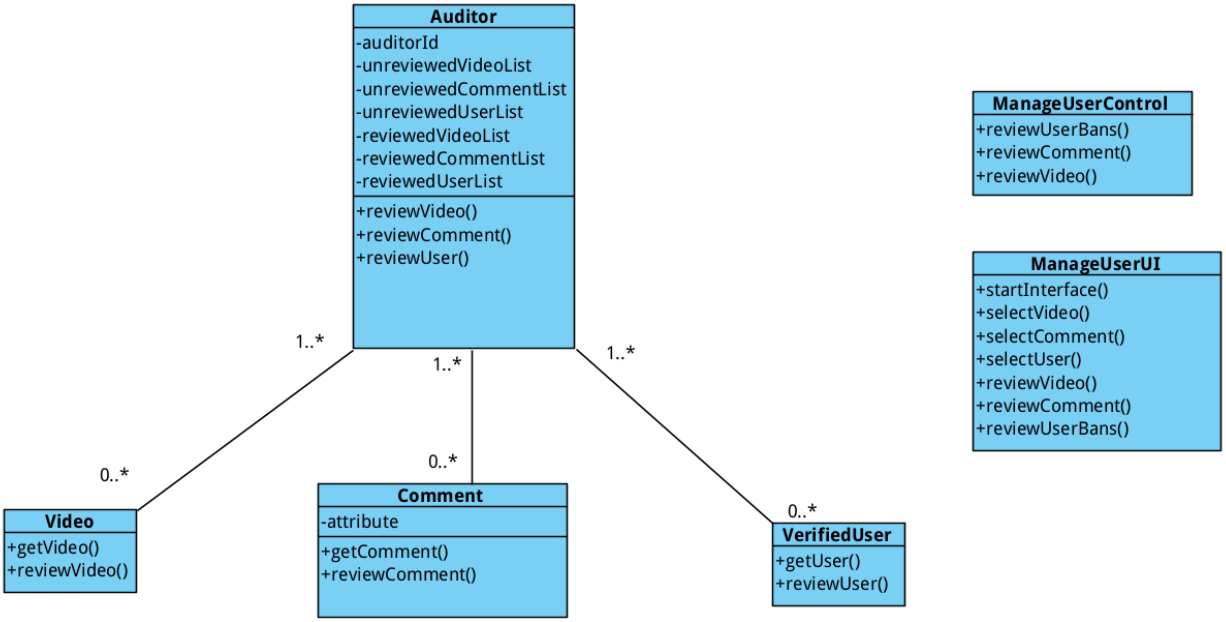






4.2.3.2 类图

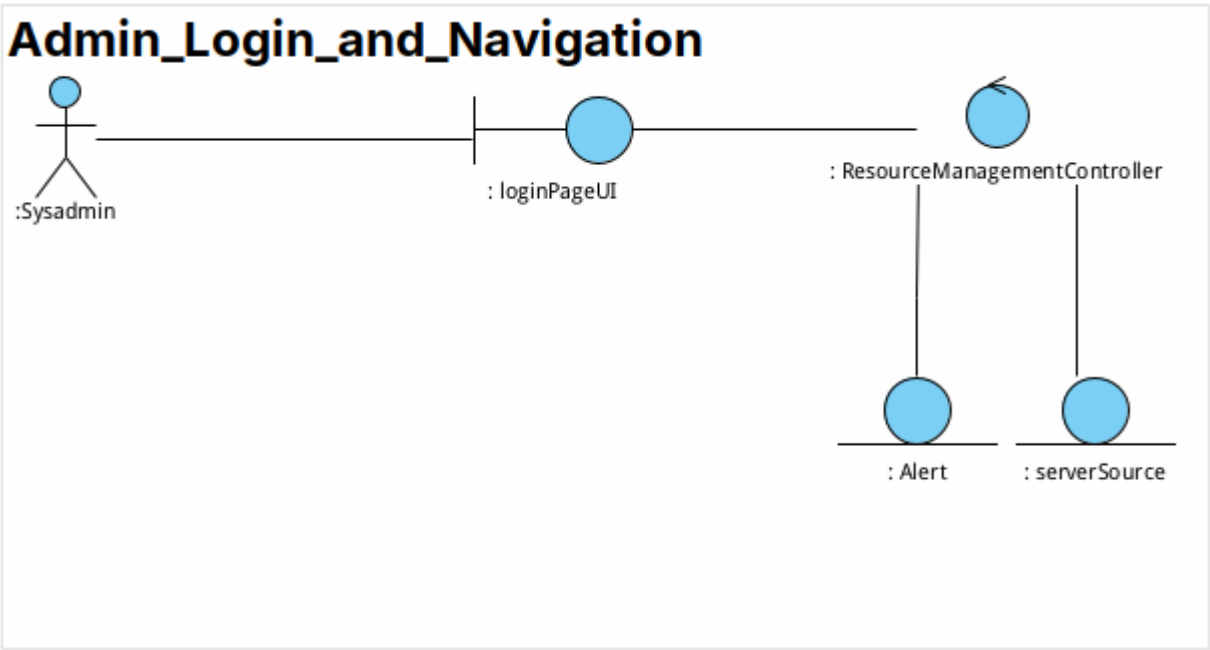
将审核员管理正式用户的通讯图转化为类图。

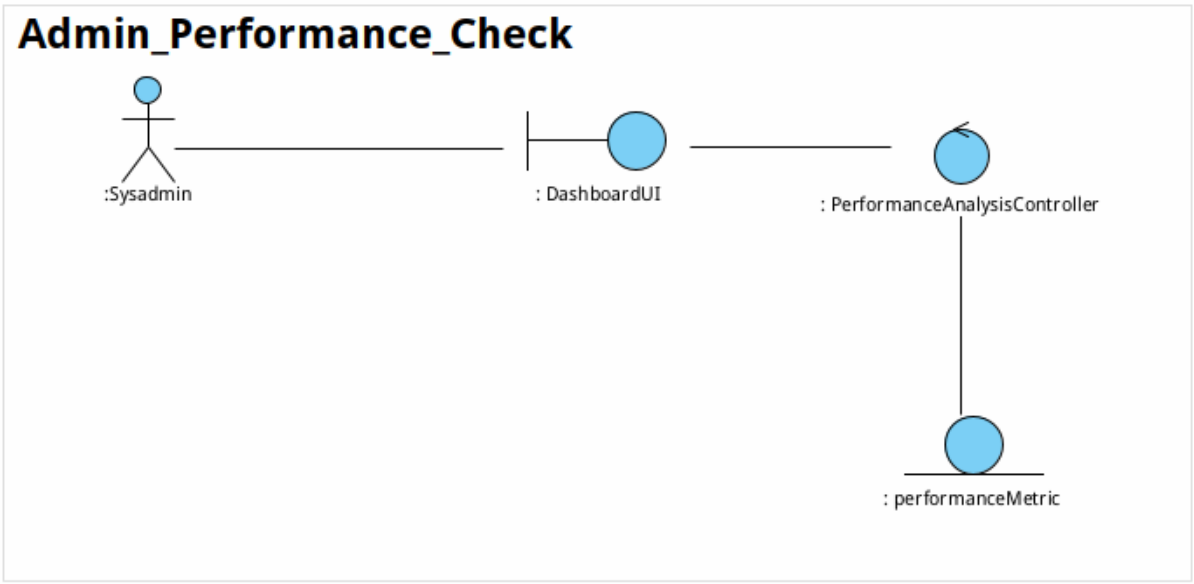


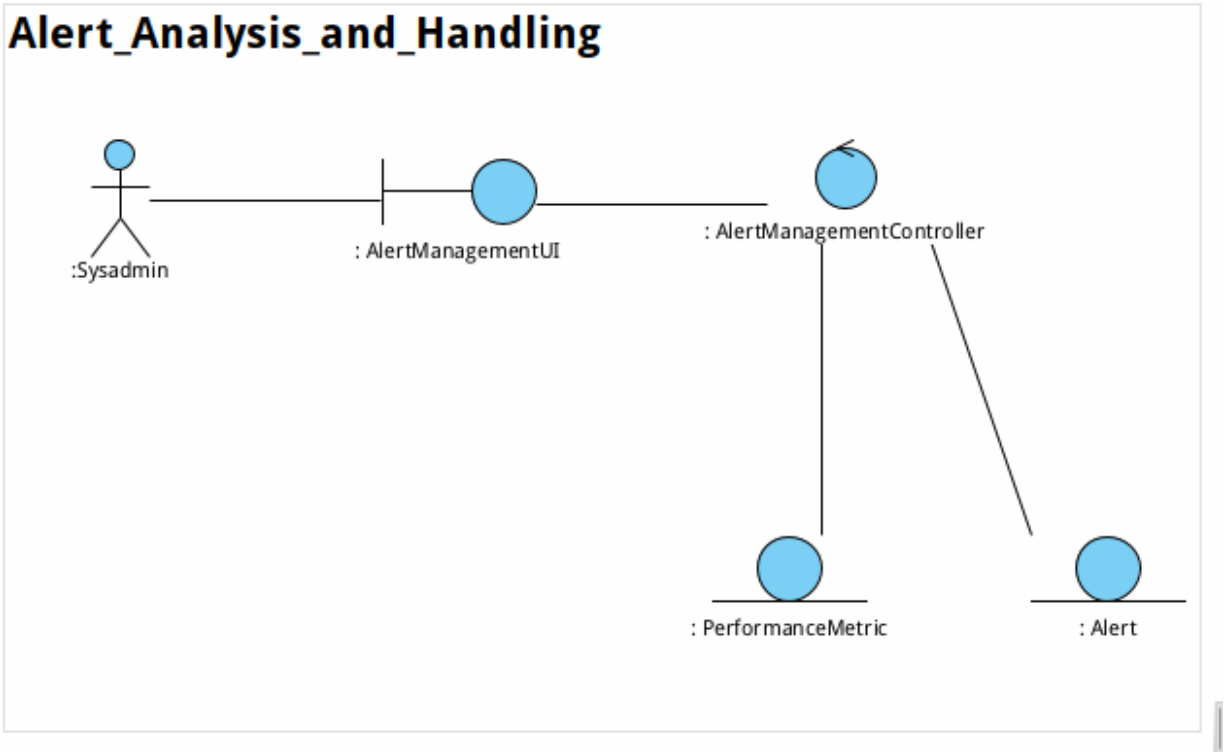
4.2.4 系统管理员

4.2.4.1 协作图

对系统管理员相关用况进行健壮性分析，已经初步确定了主要参与的实体对象，与其连接的边界类和控制类，见下图。



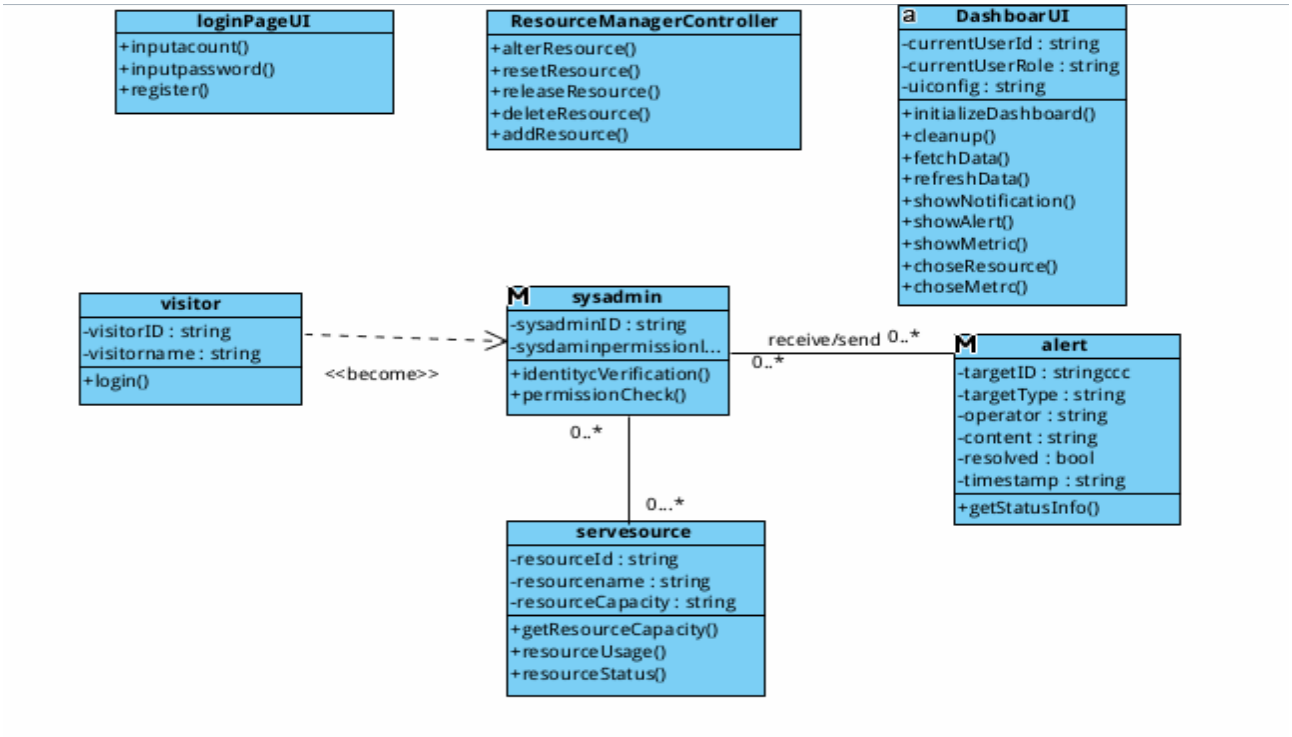




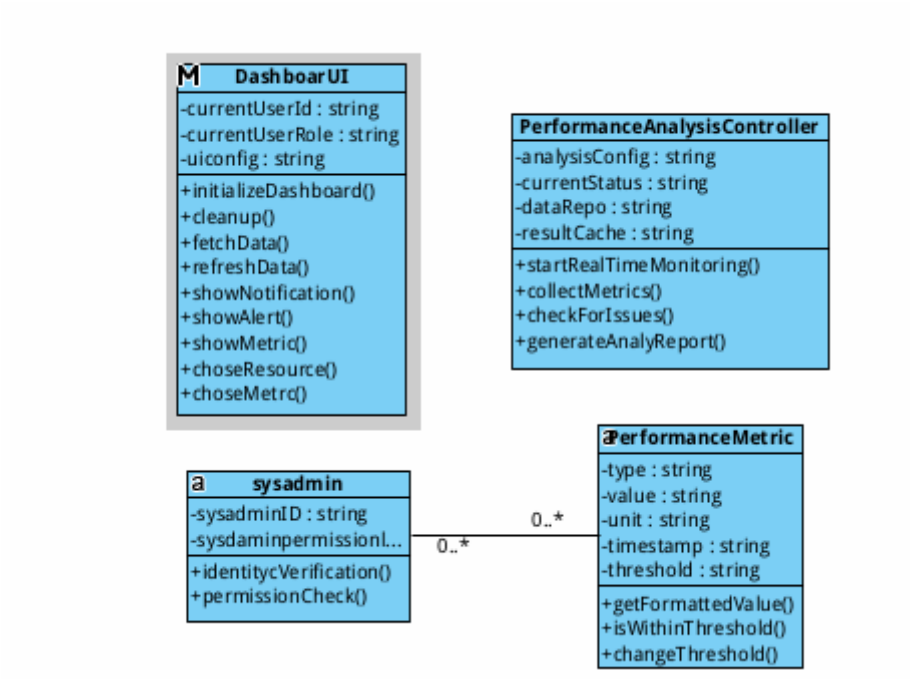
4.2.4.2 类图

系统管理员用况的类图，见下图。

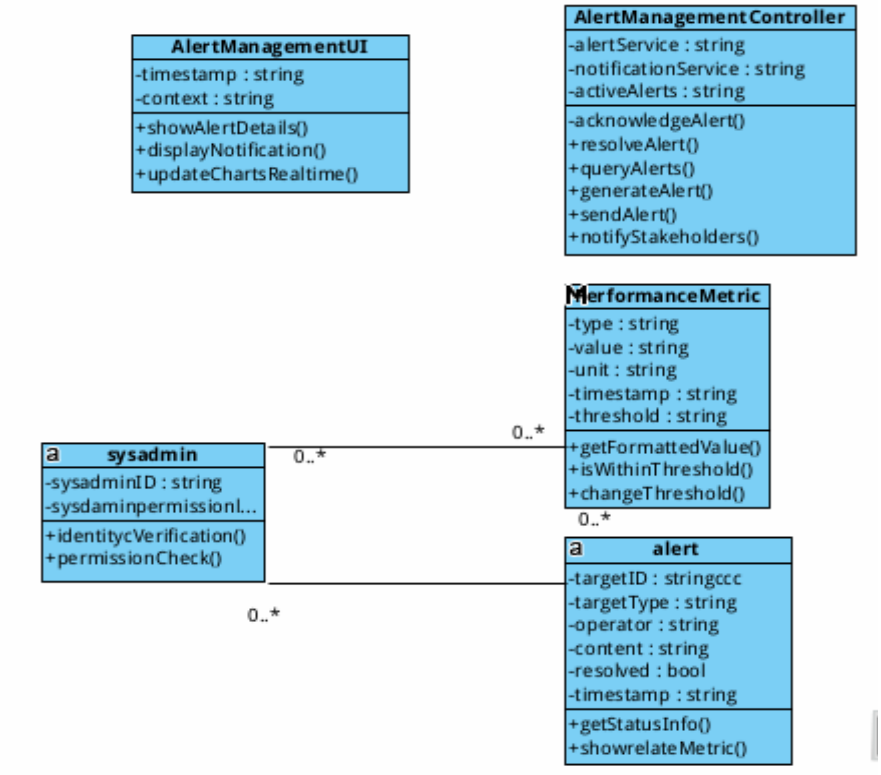
系统管理员登陆和管理资源：


loginPageUI
+inputacount()
+inputpassword()
+register()
ResourceManagerController
+alterResource()
+resetResource()
+releaseResource()
+deleteResource()
+addResource()
DashboarUI
-currentUserId : string
-currentUserRole : string
-uiconfig : string
+initializeDashboard()
+cleanup()
+fetchData()
+refreshData()
+showNotification()
+showAlert()
+showMetric()
+choseResource()
+choseMetrc()
visitor
-visitorID : string
-visitorname : string
+login()
<<become>>
sysadmin
-sysadminID : string
-sysdaminpermissionl...
+identitycVerification()
+permissionCheck()
receive/send
0..*
0..*
alert
-targetID : stringccc
-targetType : string
-operator : string
-content : string
-resolved : bool
-timestamp : string
+getStatusInfo()
0..*
0...*
servesource
-resourceId : string
-resourcename : string
-resourceCapacity : string
+getResourceCapacity()
+resourceUsage()
+resourceStatus()

系统管理员分析性能：

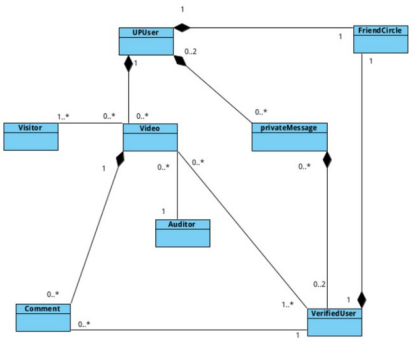


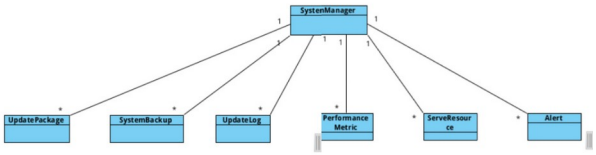
系统管理员分析和处理警告：



4.3. 总体类图

该图描述各个类之间的关系及其多重性。


UPUser
FriendCircle
Visitor
Video
privateMessage
Auditor
Comment
VerifiedUser


SystenManager
UpdatePackage
SystemBackup
UpdateLog
Performance Metric
ServeResour ce
Alert

5 架构设计

5.1. 设计目标与约束

5.1.1 设计目标

功能性目标：全面覆盖平台核心场景，包括游客浏览视频、正式用户互动交流、UP 主创作管理、审核员内容审核及系统管理员运维监控等全流程，确保各角色功能模块既独立运行又协同高效，精准落地 “高质量视频 + 实时弹幕 + 深度社群” 的核心产品定位。

性能目标：适配高并发访问场景，严格达成首页首屏加载时间＜3 秒、视频缓冲时间＜5 秒的既定指标；保障弹幕实时推送、视频流畅播放及多用户同步互动，核心服务响应延迟控制在 100ms 以内，提升用户使用体验。

可靠性目标：确保系统 7×24 小时稳定运行，服务可用性不低于 99.9%；

安全性目标：实现用户密码加密存储、敏感操作二次验证；结合自动检测与人工审核机制拦截违规内容，防范合规风险；严格划分用户权限边界，避免数据泄露与非法操作。

可维护性目标：架构设计简洁清晰，明确各服务职责边界，支持独立部署与升级；配套完善的监控与日志系统，便于快速定位和排查问题；选用成熟稳定的技术栈，降低长期维护成本。

5.1.2 设计约束

技术栈约束：基于项目既定技术选型，后端采用 C++/QML，数据库使用 MySQL，前端选用 QML，需在该技术栈范围内实现架构落地，确保开发效率与系统兼容性。

成本约束：首阶段聚焦 PC 端核心功能，成本需可控，架构设计需避免过度设计，优先满足最小可行产品需求；云服务器、存储等资源配置需兼顾性能与经济性，支持按需扩容。

合规约束：需符合法律法规及监管要求，架构中需嵌入内容审核流程、青少年保护机制，确保用户数据收集与使用合规，版权内容管理规范。

多端适配约束：预留移动端扩展接口，架构设计需考虑 PC 端与后续移动端的数据同步与功能兼容，确保多端用户体验一致，数据互通无壁垒。

用户规模约束：初期聚焦 16-28 岁 Z 世代垂直用户群体，架构需支撑初期用户规模平滑增长，同时具备向大规模用户拓展的横向扩展能力，避免架构重构。

5.2. 系统总体架构

5.2.1 架构概览图

5.2.2 核心服务划分

服务名称	服务职责	关键技术	是否可以进行并发
视频浏览与播放服务	支撑游客及正式用户的视频列表加载、关键词搜索、视频播放（含倍速、清晰度调节）功能；记录视频观看次数与观看进度；保障视频流稳定传	MySQL 数据库查询优化、CDN 加速、子流 S1（加载视频列表）、S2（播放视频）、S3（关键词搜	是，支持多用户同时加载视频列表、搜索视频及播放不同视频

	输。	索视频）实现	
用户认证与管理服务	处理游客注册成为正式用户（含直接注册、第三方登录）、用户登录验证、个人信息管理；管控用户权限（如正常用户、UP 主、审核员、系统管理员）；支持用户账户封禁与解封操作。	密码加密算法、第三方登录授权接口、子流 S4（验证消息）	是，支持多用户同时注册、登录及权限校验
互动交流服务	支撑正式用户的视频互动（点赞、投币、收藏、评论、回复）、弹幕发送与展示、私信交流、动态发布与分享功能；实时同步互动数据与消息通知。	WebSocket（弹幕实时推送）、消息队列、子流 S5（更新系统视频属性）、违规内容初步检测	是，弹幕与互动操作采用异步处理
UP 主创作管理服务	支持 UP 主视频上传、视频信息（标题、封面、关键词）编辑、稿件管理（隐藏、删除、更新）；提供视频数据统计反馈（播放量、粉丝数、互动数据）；处理视频上传后的审核队列提交。	分片上传技术、文件存储服务（如对象存储 COS）、MySQL 事务管理、视频格式校验	是，支持多 UP 主同时上传视频与管理稿件，上传任务异步处理，文件存储与业务逻辑解耦提升并发量
内容审核服务	接收 UP 主上传视频、用户发布的弹幕 / 评论 / 动态等内容，提供审核队列管理；支持自动关键词检测违规内容、人工审核操作（通过 / 驳回 / 删除）；记录违规行为与审核结果。	文本关键词匹配算法、视频内容预览技术、审核流程引擎	是，审核队列按优先级调度，自动审核与人工审核并行处理，支持多审核员同时在线操作
系统运维监控服务	支撑系统管理员的系统性能分析（服务器状态、流量峰值、核心指标监控）、系统更新（服务更新、数据库更新）、全局参数配置、用户权限管控；提供日志查询与故障预警功能。	数据库备份与回滚技术	是，性能监控与系统更新独立执行，支持管理员同时进行多维度监控与运维操作

5.3. 分层架构设计

5.3.1 表现层

表现层作为系统与用户直接交互的窗口，核心职责为接收用户指令、渲染界面内容，并将用户请求

委派给应用逻辑层。本层采用 QML 技术构建跨平台客户端，QML 的声明式语法与数据绑定特性，能够高效实现符合 Z 世代审美的动态、流畅交互界面，为弹幕互动、实时更新等核心场景提供技术支撑。

职责：专注于用户界面的渲染与用户交互事件的捕获，确保操作的即时反馈与视觉呈现的流畅性。

协作：表现层不包含业务逻辑，通过调用应用逻辑层提供的服务接口来完成业务请求，并依据返回数据更新界面状态。

5.3.2 应用逻辑层

应用逻辑层作为表现层与领域层之间的协调者，核心职责为组织业务用例的完整执行流程。本层通过定义清晰的服务（封装特定业务领域操作）与控制器（处理特定用户请求），编排并调用领域层的核心业务逻辑与基础设施层的技术能力，确保业务流程正确、高效地执行。

职责：作为业务流程的协调者与组织者，将用户请求转换为对领域对象和基础设施的调用序列，负责事务管理、权限校验、服务编排等横切关注点，但自身不实现核心业务规则。

协作：应用逻辑层接收来自表现层的请求，通过内部的服务与控制器进行协调，调用领域层的领域对象与领域服务执行业务逻辑，并委托基础设施层完成数据持久化、外部集成等操作。

5.3.3 领域层

领域层是系统的业务核心，负责封装与技术和平台无关的业务逻辑与规则。其目标是实现与外部框架、持久化和用户交互的解耦，确保业务逻辑的独立性和可测试性，从而灵活应对未来业务规则的变化。

职责：承载和表达核心业务概念、状态、规则与流程。领域层是业务复杂性的所在地，通过实体、值对象、领域服务等模式，确保业务逻辑的纯粹性、内聚性和独立性。

协作：领域层被应用逻辑层调用以执行业务操作，其本身不依赖任何外部技术细节。领域实体通过数据访问层进行持久化，并可在必要时触发领域事件，通知其他上下文。

5.4. 子系统划分与职责

社交互动子系统

核心职责：负责平台内所有用户间的互动行为，是构建社区氛围与用户粘性的关键。

关键功能：评论的发布与管理、回复、点赞/点踩；用户间的关注与粉丝关系维护；私信通信；动态的发布与展示。

管理核心：评论、回复、用户关系、私信、动态等互动数据。

观看视频子系统

核心职责：为用户提供稳定、流畅、沉浸式的视频消费体验，是平台最核心的用户侧功能。

关键功能：视频流的加载与播放、清晰度切换、播放进度控制、弹幕的实时渲染与显示、播放历史记录。

关联质量属性：直接关联“视频播放点击后开始缓冲时间<5 秒”的性能需求。

发布与管理视频子系统

核心职责：为 UP 主提供高效、便捷的视频内容创作与生命周期管理工具，是平台内容生态的供给端。

关键功能：视频文件与元数据（标题、封面等）的上传；已发布视频的编辑、更新、删除、隐藏；

视频数据（播放量、互动数）的统计与展示。

协作关系：视频上传后，会向审核子系统提交审核请求。

消息通知子系统

核心职责：作为系统的信息神经中枢，负责系统内各类事件触发的用户通知的生成与分发。

关键功能：系统公告的推送；用户相关的互动通知（如被点赞、被回复、新粉丝）；UP 主相关的流程通知（如视频审核结果）；实时聊天的信令传递。

审核子系统

核心职责：保障平台内容安全与合规，维护健康的社区环境。

关键功能：对用户生成的视频、弹幕、评论、动态等内容进行审核（自动规则与人工审核结合）；对违规用户进行封禁等处理；提供审核员工作台。

关联质量属性：支撑“人工审核+关键词自动检测”的合规性需求。

系统管理子系统

核心职责：为系统管理员提供全局性的监控、配置与管理能力，确保平台的稳定、安全运行。

关键功能：全局用户管理与权限配置；系统关键参数的设置与调整；监控系统健康状况（服务器资源、服务状态）并生成数据报表；系统更新与维护。

搜索与推荐子系统

核心职责：连接用户与内容，提升内容分发效率与用户内容发现体验。

关键功能：

搜索：基于关键词的多条件视频搜索，支持相关性排序。

推荐：基于用户行为、兴趣标签、社交关系等，生成个性化的首页视频推荐流。

关联质量属性：直接关联“首页首屏加载时间<3 秒”的性能需求。

用户管理子系统子系统

核心职责：管理平台所有用户的身份、认证与基础信息，是系统安全的基础。

关键功能：用户注册、登录、登出、第三方授权；个人基本信息（昵称、签名）的管理；密码修改与安全设置。

5.5. 架构部署

5.5.1 部署架构，硬件，全局资源与数据存储

5.5.1.1 部署架构设计

本系统采用”混合本地与云端区域共同部署架构”，结合公有云数据同步与本地信息存储，实现高可用、可

扩展的视频服务平台。

5.5.1.2 部署架构设计

1.本地配置：

采用 linux 系统 QT6 进行开发

所有使用库均为 QT6 自带库实现

本项目开发采用 C++与 javaScrip 指定复用计 t 语言与 QMl 混合开发

本项目文档均由 LiberOffice 撰写

2. 云服务资源

资源类型	规格配置	数量	用途
腾讯云服务器	2 核 1GB SSD 云硬盘 40GB DDoS 防护 DDoS 高防保险	1	用于提供线上服务与业务逻辑处理,涵盖云对象存储服务，用于视频对象及图片等文件的存储
腾讯云数据库	SQL 8.0 本地 SSD 盘 重庆 InnoDB 通用型-1 核 1000MB 内存 双节点 200 GB	1	用于数据表的存储及云 SQL 的使用

3.资源调度策略

智能 DNS 解析：基于用户地理位置选择最优接入点

动态扩缩容：根据负载自动调整计算资源

流量调度：基于健康检查和服务质量的路由决策

4.数据存储架构

类别	描述
结构化数据存储	MySQL 云数据库自动保存与存储。
非结构化数据存储	对象存储方案:使用腾讯云对象存储存储同进行存储。

5. 缓存体系

多级缓存架构:

L1: 本地缓存(Caffeine) - 应用服务器内部

L2: Redis 集群 - 分布式共享缓存

- 主从架构: 6 节点(3 主 3 从)

- 内存配置: 256GB/节点

- 持久化: RDB+AOF 混合模式

L3: CDN 缓存 – 边缘节点缓存

5.5.2 性能评估

性能指标与目标

1. 系统容量评估

指标	设计容量	峰值容量	扩容阈值
并发用户数	1000	2000	1500
视频播放 QPS	50	100	80
评论操作 QPS	1000	2000	1500
弹幕发送 QPS	1000	2000	1500
数据写入 TPS	1000	2000	1500
数据读取 QPS	10000	20000	15000

2. 响应时间目标

(1) API 响应时间:

P95 < 100ms: 核心业务接口(播放、点赞)

P95 < 200ms: 一般业务接口(评论、收藏)

P95 < 500ms: 复杂查询接口(搜索、推荐)

(2) 页面加载时间:

首屏时间: < 1.5 秒

可交互时间: < 2 秒

完全加载: < 3 秒

(3) 视频播放:

首帧时间: < 1 秒

缓冲比率: < 1%

卡顿率: < 0.5%

5.5.3 权衡优先级

1. 架构权衡

质量属性	优先级	权衡决策	影响范围
可用性	最高(10)	采用多活架构，牺牲部分一致性	全系统
性能	高(9)	缓存优先，数据最终一致	核心业务
可扩展性	高(9)	微服务化，增加运维复杂度	服务架构
安全性	高(8)	全面加密，增加计算开销	数据传输存储
成本效益	中(7)	混合云策略，优化资源使用	基础设施
开发效率	中(6)	标准化框架，学习成本	开发团队
可维护性	中(6)	完善监控，增加管理开销	运维体系

2. 具体权衡决策

决策一：一致性与可用性的权衡

选择: 优先保证可用性，接受最终一致性

实现:

- 核心交易数据: 强一致性(MySQL 主从同步)
- 用户行为数据: 最终一致性(消息队列异步处理)
- 缓存数据: 设置合理 TTL，定期刷新

影响: 极少数情况可能看到短暂数据不一致

决策二：性能与成本的权衡

选择: 性能优先，但通过技术优化控制成本

实现:

- 计算层: 使用弹性伸缩，闲时缩容
- 存储层: 数据分级存储(热/温/冷)
- CDN: 按流量计费，动态选择供应商
- 数据库: 读写分离，降低主库压力

决策三：安全性与用户体验的权衡

选择: 在关键操作加强安全，一般操作简化

实现:

- 登录/支付: 多因素认证，牺牲便利性
- 日常浏览: 简化验证，提升体验
- 敏感操作: 增加二次确认
- 数据传输: 全链路 HTTPS，增加延迟但必要

决策四：功能丰富度与系统稳定性的权衡

选择: 核心功能高稳定，创新功能允许试错

实现:

- 核心模块: 严格测试，灰度发布
- 实验功能: A/B 测试，快速迭代
- 弹幕/评论: 异步处理，允许短暂延迟
- 支付系统: 同步处理，确保实时性

3. 风险缓解策略

风险类别	可能性	影响程度	缓解措施
单点故障	低	极高	多可用区部署，自动故障转移
数据丢失	低	极高	多地备份，定期恢复演练
DDoS 攻击	中	高	多层防护，流量清洗
成本超支	中	中	预算监控，自动伸缩策略
性能下降	高	中	容量规划，性能监控预警
安全漏洞	中	高	安全扫描，漏洞奖励计划

5.5.4 指定复用计划

1. 技术组件复用计划

基础框架层复用

通用技术栈:

- 开发框架: Spring Boot 3.x (统一后端技术栈)
- 前端框架: React 18 + TypeScript (统一前端架构)
- 微服务治理: Spring Cloud Alibaba (服务发现、配置中心)
- 容器编排: Kubernetes (统一部署平台)
- 监控体系: Prometheus + Grafana + ELK (统一可观测性)

标准化组件库:

- 认证授权组件: OAuth2/JWT 统一解决方案
- 分布式锁组件: Redis Redlock 实现
- 消息队列组件: RocketMQ 标准化接入
- 缓存组件: 多级缓存统一封装
- 数据库访问: MyBatis Plus + ShardingSphere

业务组件复用：

视频处理流水线:

- 转码服务: FFmpeg 标准化封装
- 审核服务: 内容安全统一接口
- 分发服务: CDN 统一管理平台
- 元数据管理: 标准化视频元数据模型

用户服务体系:

- 用户中心: 统一用户信息管理
- 消息中心: 站内信/推送统一服务
- 关系系统: 关注/粉丝通用模块
- 钱包系统: 虚拟货币统一管理

内容管理体系:

- 投稿系统: 标准化投稿流程
- 审核系统: 多级审核工作流
- 推荐引擎: 个性化推荐算法框架
- 搜索服务: 全文检索引擎封装

2. 架构模式复用

设计模式库

已确认可复用的模式:

1. CQRS 模式: 适用于读写分离场景
2. Saga 模式: 分布式事务管理
3. Circuit Breaker: 服务熔断降级
4. Bulkhead: 资源隔离
5. API Gateway: 统一 API 入口

模式实现规范:

- 代码模板: 提供标准实现代码
- 配置规范: 统一配置格式
- 监控指标: 标准监控埋点
- 文档模板: 使用和扩展说明

部署模式复用

标准化部署模板:

- Docker 镜像规范: 基础镜像、构建脚本
- Kubernetes 配置: Deployment/Service/Ingress 模板
- 健康检查: 就绪探针、存活探针标准
- 资源限制: CPU/内存请求与限制规范

CI/CD 流水线:
- 代码扫描: SonarQube 统一规则
- 单元测试: 覆盖率要求(>80%)
- 集成测试: 标准测试环境
- 部署策略: 蓝绿部署/金丝雀发布模板

3. 知识资产复用
文档体系
技术文档库:
- 架构决策记录(ADR): 关键决策文档化
- API 文档: OpenAPI/Swagger 标准化
- 部署手册: 标准操作流程(SOP)
- 故障处理: 应急预案手册
- 性能调优: 最佳实践集

培训材料:
- 新员工培训: 标准技术栈培训
- 架构培训: 系统架构讲解材料
- 运维培训: 日常运维操作指南
- 安全培训: 安全开发规范

工具链复用
开发工具链:
- IDE 配置: 统一开发环境配置
- 代码规范: Checkstyle/PMD 规则集
- Git 规范: 分支策略、提交规范
- 构建工具: Maven/Gradle 统一配置

运维工具链:

- 监控告警: 统一告警规则和接收人
- 日志分析: 标准日志格式和检索
- 配置管理: 统一配置管理平台
- 密钥管理: 密钥存储和轮换机制

4. 复用治理机制

复用评审流程

新组件开发 → 架构评审委员会 → 标准化评估 →

├── 通过 → 加入复用库 → 文档化 → 推广使用

└── 不通过 → 项目内使用 → 后续评估

复用质量保障

质量门禁:

- 代码质量: SonarQube 扫描必须通过
- 测试覆盖: 单元测试覆盖率>80%
- 文档完整: API 文档、使用示例齐全
- 性能基准: 性能测试报告达标
- 安全扫描: 无高危安全漏洞

维护机制:

- 版本管理: 语义化版本控制
- 兼容性: 向后兼容性保证
- 废弃策略: 提前通知、迁移指导
- 问题响应: SLA 保障响应时间

复用度度量指标

指标名称	目标值	测量方法	改进措施
组件复用率	>60%	统计复用组件数量占比	建设组件库，降低使用门槛
代码重复率	<5%	静态代码分析工具	提取公共组件，重构重复代码
新项目启动时间	<3 天	从立项到开发环境就绪	完善脚手架和模板
故障平均修复时间	<1 小时	监控系统记录	完善监控和自动化恢复
开发人员满意度	>4.0/5.0	季度调查问卷	持续改进开发体验

通过系统化的复用计划，我们期望达到以下目标：

1. 新功能开发效率提升 40%
2. 系统维护成本降低 30%

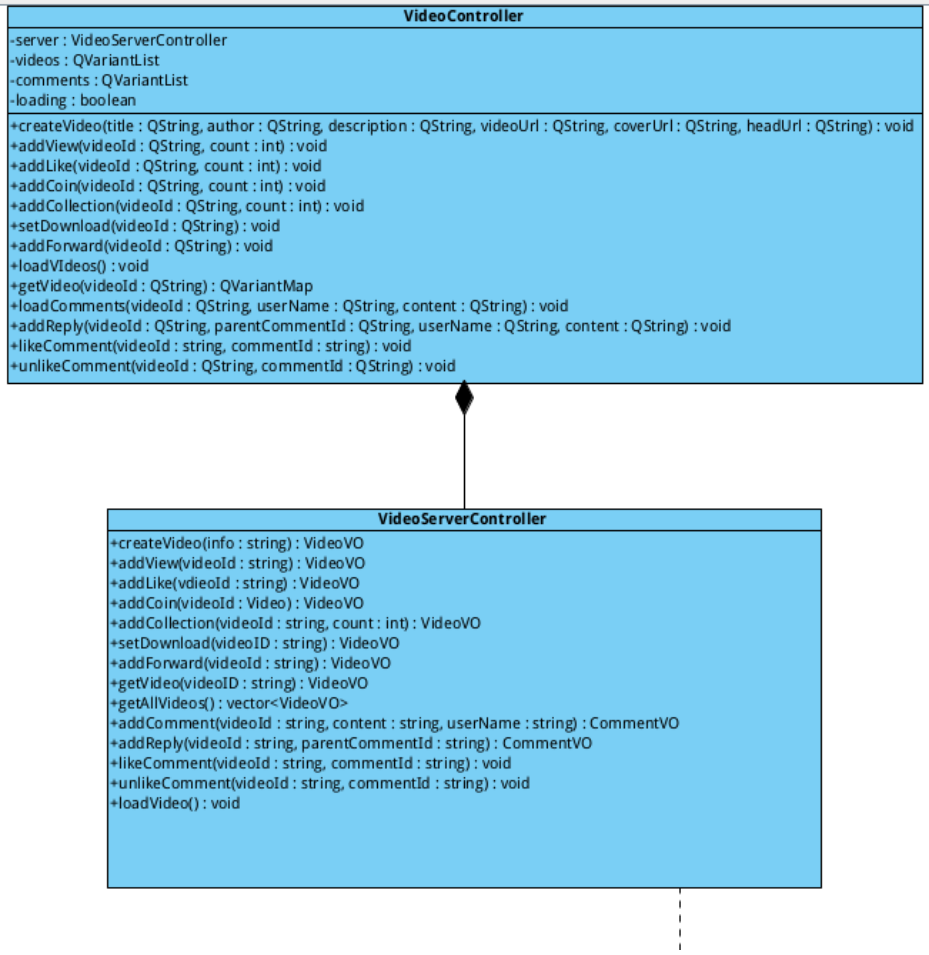
3. 技术债务增长率控制在 5%以内

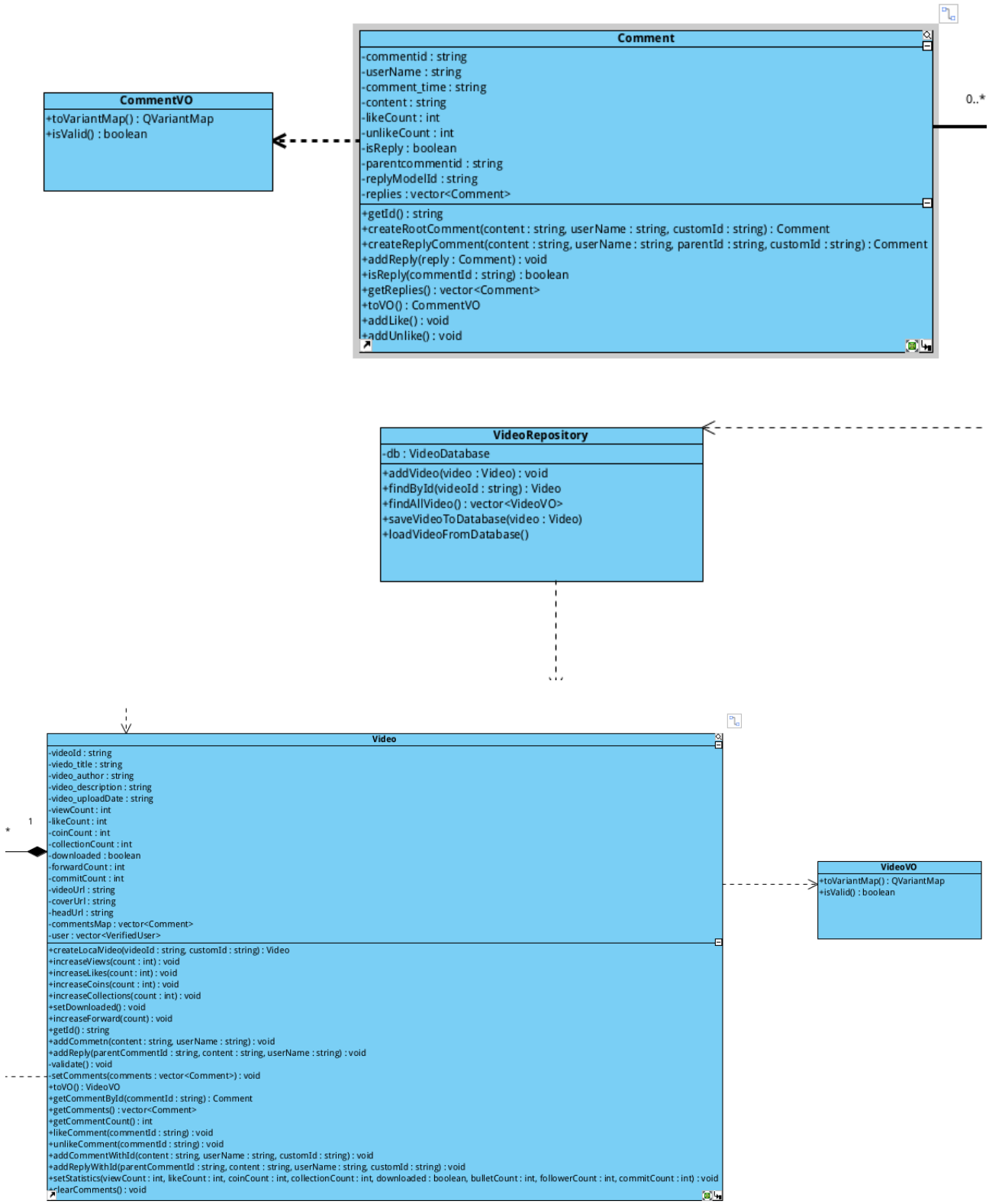
4. 团队知识共享度达到 80%

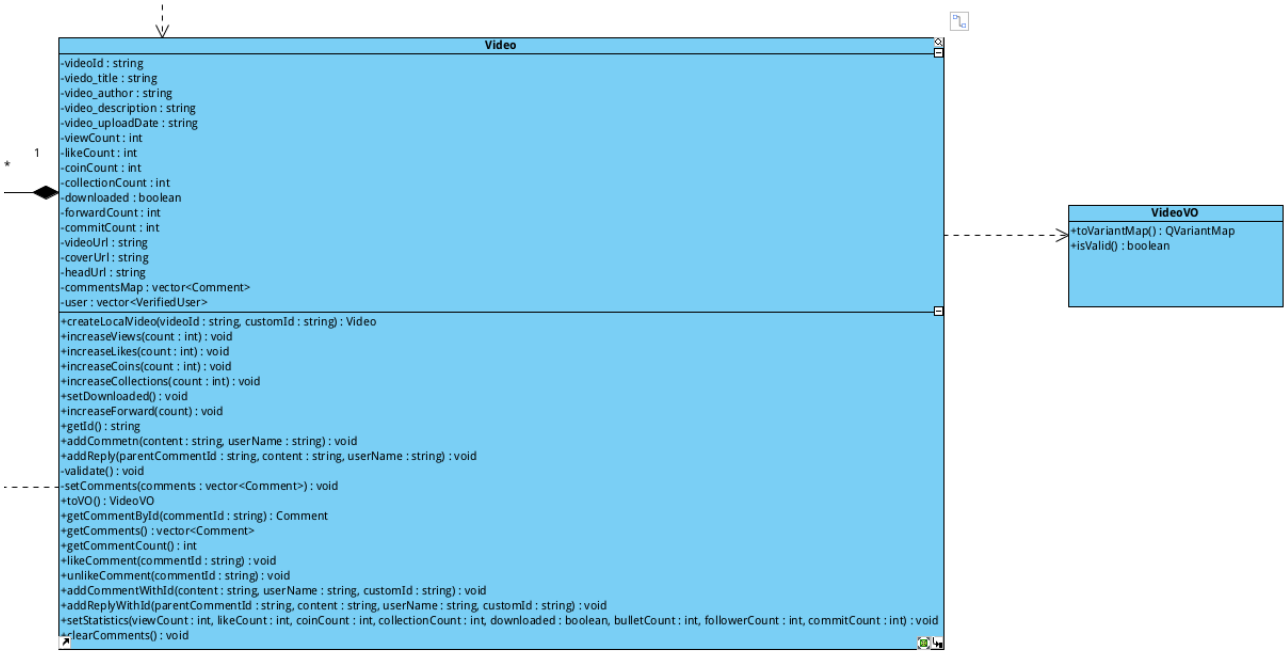
5. 整体系统稳定性提升到 99.99%

6 详细设计

6.1. 观看视频并互动用况详细类图




Comment
-commentid : string
-userName : string
-comment_time : string
-content : string
-likeCount : int
-unlikeCount : int
-isReply : boolean
-parentcommentid : string
-replyModelId : string
-replies : vector<Comment>
+getId() : string
+createRootComment(content : string, userName : string, customId : string) : Comment
+createReplyComment(content : string, userName : string, parentId : string, customId : string) : Comment
+addReply(reply : Comment) : void
+isReply(commentId : string) : boolean
+getReplies() : vector<Comment>
+toVO() : CommentVO
+addLike() : void
+addUnlike() : void
0..*
CommentVO
+toVariantMap() : QVariantMap
+isValid() : boolean
VideoRepository
-db : VideoDatabase
+addVideo(video : Video) : void
+findById(videoId : string) : Video
+findAllVideo() : vector<VideoVO>
+saveVideoToDatabase(video : Video)
+loadVideoFromDatabase()
Video
-videoId : string
-viedo_title : string
-video_author : string
-video_description : string
-video_uploadDate : string
-viewCount : int
-likeCount : int
-coinCount : int
-collectionCount : int
-downloaded : boolean
-forwardCount : int
-commitCount : int
-videoUrl : string
-coverUrl : string
-headUrl : string
-commentsMap : vector<Comment>
-user : vector<VerifiedUser>
+createLocalVideo(videoId : string, customId : string) : Video
+increaseViews(count : int) : void
+increaseLikes(count : int) : void
+increaseCoins(count : int) : void
+increaseCollections(count : int) : void
+setDownloaded() : void
+increaseForward(count) : void
+getId() : string
+addCommetn(content : string, userName : string) : void
+addReply(parentCommentId : string, content : string, userName : string) : void
-validate() : void
-setComments(comments : vector<Comment>) : void
+toVO() : VideoVO
+getCommentById(commentId : string) : Comment
+getComments() : vector<Comment>
+getCommentCount() : int
+likeComment(commentId : string) : void
+unlikeComment(commentId : string) : void
+addCommentWithId(content : string, userName : string, customId : string) : void
+addReplyWithId(parentCommentId : string, content : string, userName : string, customId : string) : void
+setStatistics(viewCount : int, likeCount : int, coinCount : int, collectionCount : int, downloaded : boolean, bulletCount : int, followerCount : int, commitCount : int) : void
-clearComments() : void
1
*
VideoVO
+toVariantMap() : QVariantMap
+isValid() : boolean



描述：根据四层架构分为：

1.表现层：VideoController

2.应用层：VideoServerController

3.领域层：Video、VideoRepository、VideoVO、Comment、CommentVO

4.数据层：VideoDatabase

表现层与前端 qml 相连，负责同步前端显示并且和后端连接，接受用户操作后会通过表现层的 controller 与应用层建立连接执行后端逻辑操作。

应用层通过协调领域层，组建领域层的对象进行业务逻辑流程。

领域层中，video、comment 作为核心类，执行具体的业务操作，如点赞、投币、创建视频等;videoVO、commentVO 作为数据传输，在不暴露 video 内部数据的情况下，向 qml 提供数据;VideoRepository 作为仓存储，但需要保持到数据库或者拉取数据的时候，通过该对象去获取。

数据层 VideoDatabase 主要用于将数据保持到数据库，获取数据从数据库，同时支持将数据拉取后加载到本地内存中。

6.2. 软件控制策略、边界条件处理、并发性

6.2.1 软件控制策略

(1)分层架构控制策略

表现层 (Presentation Layer):

- 策略: 客户端负载均衡 + 服务端渲染(SSR)混合

- 实现: Next.js/Vue Nuxt 框架 + CDN 静态资源分发

- 作用: 分离前后端关注点，减轻服务器压力

应用层 (Application Layer):

- 策略: 基于 Spring Cloud/Alibaba 的微服务架构
- 实现:
  - 服务注册发现: Nacos/Consul
  - 配置中心: Apollo/Nacos Config
  - API 网关: Spring Cloud Gateway
- 作用: 业务逻辑编排，服务治理

领域层 (Domain Layer):

- 策略: 领域驱动设计(DDD) + CQRS 模式
- 实现:
  - 写模型: 事务性领域服务
  - 读模型: 独立查询服务+Redis 缓存
- 作用: 业务逻辑核心，保障数据一致性

基础设施层 (Infrastructure Layer):

- 策略: 抽象化+依赖倒置
- 实现: 仓储模式，统一数据访问接口
- 作用: 技术细节隔离，便于技术栈更换

(2)核心业务控制策略

视频处理流程：采用状态机模式控制转码、审核、发布流程

评论系统：采用责任链模式进行敏感词过滤、垃圾评论识别

推荐算法：采用策略模式支持多算法动态切换（协同过滤、内容推荐、深度学习）

支付系统：采用事务脚本模式确保资金操作原子性

消息通知：采用观察者模式实现多通道通知（站内信、邮件、App 推送）

6.2.2 边界条件处理

[1]业务边界条件

业务场景	边界条件	处理策略
用户注册	手机号/邮箱重复	数据库唯一索引 + Redis 分布式锁
视频发布	审核中状态并发修改	乐观锁(version 字段) + 状态机验证

业务场景	边界条件	处理策略
弹幕发送	发送频率限制	滑动窗口限流(Redis + Lua 脚本)
评论删除	级联删除权限验证	RBAC 权限检查 + 异步清理任务
支付回调	重复通知处理	幂等性设计(支付单号+状态机)

[2]系统边界条件

服务依赖降级：当推荐服务不可用时，自动切换为热度排序

第三方 API 容错：短信/邮件服务失败时，存入消息队列重试

地理区域限制：基于 IP 识别，限制特定区域的内容访问

时间窗口限制：限制特定时段的高消耗操作（如大规模数据导出）

6.2.3 并发性处理设计

在本系统中，并发性处理是保障高可用、高性能和最终用户体验的核心。我们主要面临**读高并发**和**写高并发**两种场景，并通过分层、异步、最终一致性等架构理念进行系统化应对。

一、整体并发架构原则

1. **读写分离与缓存先行**：所有读请求路径上设置多层缓存，数据库主库仅处理写事务和核心实时读。写操作通过异步机制同步至读库与缓存，保证读路径的极高响应速度。

2. **服务无状态与横向扩展**：应用服务层不存储会话状态（用户状态存储于 Redis），便于通过增加 Pod 或实例数量水平扩展，轻松应对流量洪峰。

3. **异步化与解耦**：非实时必需的操作（如计数更新、消息推送、日志记录）均通过消息队列（如 Kafka/RocketMQ）异步处理，将即时响应与后台处理解耦，削峰填谷。

4. **数据分区与分片**：对用户、视频、评论等核心数据依据 ID 或业务键进行分库分表，将集中式的并发压力分散到多个数据库实例。

二、核心高并发场景及解决方案

场景一：视频播放与热点内容访问（读高并发）

- **挑战**：热门视频、首页推荐流、热门直播等场景，会产生海量的瞬时读请求。
- **解决方案**：
    - **四级缓存加速体系**：

        1. **客户端本地缓存**：对静态资源（如封面图、图标）使用强缓存（`Cache-Control`）。

        2. **CDN 边缘缓存**：视频切片文件、静态页面通过 CDN 全球分发，从离用户最近的节点获取。

        3. **应用层分布式缓存（Redis Cluster）**：缓存视频元数据（标题、作者、基础计数）、热门评论列表、用户会话信息。采用旁路缓存策略，失效时由应用回源数据库并重新填充。

        4. **数据库从库缓存**：MySQL 使用 InnoDB Buffer Pool 缓存热点数据页。

    - **请求合并与防穿透/击穿**：对于瞬时失效的极端热点 Key（如顶级主播直播间信息），

使用 Redis 分布式互斥锁或“逻辑过期”标记，仅允许一个请求回源加载，其他请求短暂等待后读取新缓存，避免集体击穿数据库。

场景二：互动行为（点赞、投币、收藏）与计数更新（写高并发）

- **挑战**：短时间内对同一视频/评论的大量写操作（如“一键三连”），需保证计数准确、用户行为不重复、响应迅速。
- **解决方案**：
  - **Redis 原子操作与 Set 去重**：用户点击“点赞”时，首先在 Redis 中执行 `SADD video:liked_users:{videoId} {userId}`，利用 Set 唯一性判断是否已点。同时，使用 `HINCRBY` 原子递增计数哈希表 `video:stats:{videoId}` 中的 `like_count` 字段。
  - **异步持久化**：计数的增量变化通过消息队列发送给后台 worker 服务，worker 进行批量合并后（如每累计 100 次或每 5 秒）再更新 MySQL 数据库。这确保了前端响应毫秒级完成，同时保证数据的最终一致性。
  - **定时对账**：设置离线对账任务，定期比对 Redis 中的计数总和与 MySQL 中的持久化值，修正因极端情况（如服务重启）导致的不一致。

场景三：实时弹幕与聊天（超高并发实时交互）

- **挑战**：大型直播间的弹幕洪峰，要求毫秒级广播给数万甚至数十万在线连接。
- **解决方案**：
  - **WebSocket 长连接集群**：使用 Netty 等高性能框架搭建 WebSocket 服务器集群，承载用户长连接。
  - **连接会话管理**：用户连接建立后，其与房间的映射关系注册到 Redis 中。网关层根据用户订阅的房间 ID，将连接路由到对应的 WebSocket 服务器实例。
  - **消息广播**：
    - 用户发送弹幕 → 业务服务处理（敏感词过滤、频率限制） → 将弹幕消息发布到对应房间的 Kafka Topic。
    - 订阅了该房间 Topic 的所有 WebSocket 服务器实例消费消息 → 向各自持有的、属于该房间的用户连接进行推送。
  - **历史弹幕存储**：弹幕同时写入按房间 ID 和小时分片的 NoSQL 数据库（如 HBase）或时序数据库，用于支持“查看历史弹幕”功能。

场景四：秒杀活动（限时抢购、限量礼品码）（极端读写混合高并发）

- **挑战**：瞬间涌入远超库存数量的请求，需防止超卖、保证公平性、避免系统崩溃。
- **解决方案（令牌桶+队列化）**：

1. **请求准入与限流**：在网关层对活动接口进行毫秒级滑动窗口限流，将 99%的无效或超量请求快速拒绝，保护下游服务。

2. **库存预热与原子扣减**：活动开始前，将商品库存（如 1000 个）预存到 Redis 中。通过 `DECR` 或 Lua 脚本保证扣减的原子性，扣至 0 后直接返回“已售罄”。

3. **订单排队与异步创建**：成功预扣库存的用户，系统生成一个“排队中”的订单号，并立即返回给用户。详细的订单创建、支付链接生成等复杂操作推入消息队列，由订单服务异步顺序处理。用户可通过订单号轮询查询最终状态。

三、数据库层面的并发控制

- **乐观锁**：适用于冲突较少的场景，如在更新视频信息（标题、简介）时，使用 `version` 字段或更新时间戳 `updated_at` 进行条件更新，防止更新覆盖。
- **事务隔离级别**：默认使用 `REPEATABLE READ`（MySQL InnoDB 默认级别），在特定业务场景（如对账）可降级为 `READ COMMITTED` 以提高并发性能。

四、分布式锁的应用

使用 **Redisson** 实现的 Redis 分布式锁，用于保证跨服务或跨 JVM 进程的临界区操作互斥。

- **应用场景**：
  - 同一用户只能在一处设备登录，踢出旧会话。
  - 全局定时任务（如生成每日热门榜）的唯一起动。
  - 防止缓存重建时的并发击穿（可与“逻辑过期”方案互补）。
- **最佳实践**：锁命名具备业务含义，设置合理的自动过期时间，并实现锁续期和可重入逻辑，避免死锁。

五、监控与弹性伸缩

- **监控指标**：密切监控各层级的并发指标，包括：网关 QPS、服务实例的 CPU/内存/线程池状态、Redis 连接数与内存使用率、数据库活跃连接数与慢查询率、消息队列堆积量。
- **弹性伸缩**：
  - **横向伸缩**：基于 CPU 使用率、请求 QPS 等指标，配置 Kubernetes HPA 或云服务商的自动伸缩组，自动增减无状态服务实例。
  - **缓存与数据库扩容**：Redis Cluster 支持在线增加分片。数据库采用读写分离，并可在业务低峰期进行分库分表扩容。

6.3. 用况转化输入输出

6.3.1 视频消费与互动者用况

6.3.1.1 观看视频并互动

Video

名字 属性	videoId string	video_title string	video_description string	video_uploadDate string	video_author string
名字 属性	viewCount int	likeCount int	coinCount int	collectionCount int	downloaded bool
名字 属性	forwardCount int	commitCount int	videoUrl string	coverUrl string	comments Comment
名字 属性	user VerifiefUser				
方法	play（videoId）	pause（）	loadVideo()	public(videoInfo)	addView()

	addLike()	addForward()	addCoin()	approve(video)	reject(video)

Comment

名字	commentId	userName	comment_time	content	likeCount
属性	string	string	string	string	int

名字	parentcommentid	replies	video	user	
属性	string	Comment	Video	VerifiedUser	
方法	sendComment()	loadComment()	deleteComment()	likeComment()	replyComment(commentId)

在播放视频和互动情况下：

输入来源	输入内容	处理输入	输出
用户	用户界面上的视频选择操作（如点击视频封面），携带目标视频的唯一标识符（videoId）。	1.前端界面捕获点击事件，将 videoId 发送至视频流服务 2.系统校验当前用户权限 3.视频服务根据 `videoId` 从数据库或缓存中查询视频元数据（标题、作者、时长）及视频流地址（`playUrl`）。 4.异步记录一次播放请求。若为用户，则更新其观看历史（记录 `videoId` 和初始进度 0）。 5.并行请求评论服务，加载该视频下的评论列表。	1.成功输出：视频流数据开始传输至播放器并开始播放；视频元数据与评论列表渲染在界面上。输出目标：`VideoPlayerUI`（播放器组件）、`VideoInfoPanel`（信息面板）、`CommentSection`（评论区域）。 2.失败输出：如视频不存在或无权访问，系统返回错误信息（如“视频无法播放”）。输出目标：`MainpageUI`（提示信息）。
用户	互动类型、目标视频 ID（`videoId`）、用户 ID	1.身份与状态校验：互动控制器	成功输出：更新后的点赞数、用户当前点赞状态（如“已赞”）。输出目标：`VideoPlayerUI`（更新点赞计数器和按钮状态）。

	（userId）。	（InteractionController）验证用户状态是否正常，并查询该用户对当前视频的点赞状态。 2.执行领域逻辑(以点赞为例): 点赞：若未点赞，调用Video实体的addLike(userId)方法，视频的likeCount原子性加1，并在用户互动记录中标记。可选通知视频作者。 取消点赞：若已点赞，执行逆操作，计数减1，移除标记。 3.持久化：将更新后的点赞数和用户互动状态保存至数据库。	失败输出：如用户未登录（游客），提示“请登录后操作”；如网络异常，提示“操作失败”。 输出目标：VideoPlayerUI（提示信息）。
正式用户	评论内容（content，可含文本、表情）、videoId、userId、parentCommentId（此处为null，表示顶级评论）。	1.内容初审：评论控制器（CommentController）对内容进行快速安全校验（如敏感词检测）。 2.创建评论实体：校验通过后，调用Comment实体的createComment(content, authorId, videoId, null)工厂方法或构造函数，生	成功输出：新创建的评论数据（含commentId, content, authorInfo, createTime）。输出目标：CommentSection（将新评论插入评论列表顶部）。 失败输出：如内容违规被初审拦截，提示“评论内容包含违规信息”。 输出目标：评论输入框附近（提示信息）。

		成新的 Comment 实例，初始化 likeCount 等属性。 3.持久化与关联：将新评论保存至数据库。原子性增加对应 Video 实体的 commentCount 。 4.异步处理：评论进入审核队列（若为先审后发模式，则状态为“待审核”）。通过 WebSocket 实时推送新评论给其他在线观看者。	
正式用户	回复内容（content）、videoId、userId、parentCommentId（被回复评论的 ID）。	1.流程同发表评论：处理流程与发表顶级评论基本一致。 2.关键区别：创建的 Comment 实例中，parentCommentId 字段被赋值为被回复评论的 ID 3.通知：系统生成通知，提醒被回复的用户（parentComment 的作者）。	成功输出：新创建的回复数据。 输出目标：CommentSection（将回复插入到对应评论的线程中）。 失败输出：同发表评论。

6.3.1.2 观看视频并互动

UI 层-PrivateMessageUI

类名	PrivateMessageUI
功能定位	前端界面交互入口，负责私信界面的展示、私信发送触发及启动流程控制
关联方法	+showPrivateMessage()

类名	PrivateMessageUI
	+sendPrivateMessage() +startInterface()
输入来源	
输入内容	
约束条件	无数据格式校验（校验由 PrivateMessageController 处理）
输出目标	
输出内容	Bool + String（发送成功/失败 + 提示语，如 *“私信发送成功”*）
输出场景	
处理步骤关联	startInterface()→ 渲染私信界面 触发 sendPrivateMessage()→ 传递“发送”操作事件至 PrivateMessageController

UI 层 - FriendCircleUI

类名	FriendCircleUI
功能定位	前端界面交互入口，负责朋友圈界面展示、发布/点赞触发及启动流程控制
关联方法	+showFriendCircle() +publishFriendCircle() +startInterface() +likeFriendCircle()
输入来源	
输入内容	
约束条件	无数据格式校验（校验由 FriendCircleController 处理）
输出目标	
输出内容	Bool + String（操作成功/失败 + 提示语，如 *“朋友圈发布成功”*）
输出场景	
处理步骤关联	1.startInterface()→ 渲染朋友圈界面 2. 触发 publishFriendCircle()→ 传递“发布内容”至 FriendCircleController 3. 触发 likeFriendCircle()→ 传递“点赞目标”至 FriendCircleController

Controller 层 - PrivateMessageController

类名	PrivateMessageController
功能定位	业务逻辑核心，负责收发流程控制、消息存储与历史查询
关联属性	- userInbox : unordered_map<string, vector<PrivateMessage*>>

类名	PrivateMessageController
关联方法	- userOutbox : unordered_map<string, vector<PrivateMessage*>> - messageStore : unordered_map<string, PrivateMessage> +getPrivateMessageHistory(senderId:string, receiverId:string) : PrivateMessage[] +sendPrivateMessage(content:string, senderId:string, receiverId:string) : boolean
输入来源	PrivateMessageUI）的操作请求、系统内部消息触发（如定时任务，若需扩展）
输入内容	senderId/receiverId 字符串、content 字符串；“发送请求事件”“查询请求事件”）
约束条件	1.senderId/receiverId 需为系统中已存在的 VerifiedUser.userId 2. content 非空（长度规则由 UI 层前置校验或 Controller 二次校验） 3. 消息存储需保证原子性（如多线程场景下的锁机制）
输出目标	PrivateMessageUI）的结果反馈、数据存储层的消息持久化（需关联数据库操作类，图中未显式体现则默认内部处理）
输出内容	Bool + String（操作成功/失败 + 提示语，如 *“私信发送成功”*） PrivateMessage[]（私信历史列表，用于界面渲染）
输出场景	
处理步骤关联	sendPrivateMessage 请求 → 解析参数 → 校验合法性 → 创建 PrivateMessage 实例 → 写入 userOutbox/sender+ userInbox/receiver→ 持久化至 messageStore→ 反馈 UI 层结果 接收 getPrivateMessageHistory 请求 → 解析 senderId/receiverId→ 从 userInbox/receiver 查询消息 → 封装为数组 → 反馈 UI 层

Controller 层 - FriendCircleController

类名	FriendCircleController
功能定位	业务逻辑核心，负责发布流程控制、历史查询及审核关联
关联属性	- userCircles : unordered_map<string, vector<FriendCircle*>> - circleStore : unordered_map<string, FriendCircle>
关联方法	+getAllFriendCircleHistory(userId:string) : vector<FriendCircle> +publishFriendCircle(content:string, publisherId:string) : boolean
输入来源	FriendCircleUI）的操作请求、审核模块（Auditor 相关逻辑，若耦合）
输入内容	userId 字符串、content 字符串；“发布请求事件”“查询请求事件”）
约束条件	1.publisherId 需为系统中已存在的 VerifiedUser.userId 2. content 非空（长度规则由 UI 层前置校验或 Controller 二次校验） 3. 发布时需触发审核流程（调用 FriendCircle.checkFriendCircle()或关联 Auditor 逻辑）
输出目标	FriendCircleUI）的结果反馈、数据存储层的朋友圈持久化（需关联数据库操作类，图中未显式体现则默认内部处理）

类名	FriendCircleController
输出内容	Bool + String（操作成功/失败 + 提示语，如 *“朋友圈发布成功”*） vector<FriendCircle>（朋友圈历史列表，用于界面渲染）
输出场景	
处理步骤关联	1.publishFriendCircle 请求 → 解析参数 → 校验合法性 → 创建 FriendCircle 实例 → 调用 FriendCircle.checkFriendCircle()审核 → 写入 userCircles/publisher+ circleStore→ 然后加进数据库持久化 → 反馈 UI 层结果 2. 接收 getAllFriendCircleHistory 请求 → 解析 userId→ 从 userCircles/user 查询朋友圈 → 封装为列表 → 反馈 UI 层

Model 层 - FriendCircle（实体类）

字段名	数据类型
message_id	VARCHAR(64)
sender_id	VARCHAR(64)
receiver_id	VARCHAR(64)
content	TEXT
read_status	BOOLEAN
sent_at	TIMESTAMP
is_deleted	BOOLEAN

字段名	数据类型
message_id	VARCHAR(64)
sender_id	VARCHAR(64)
receiver_id	VARCHAR(64)
content	TEXT

Model 层 - Auditor（实体类）

类名	Auditor
功能定位	实体模型，标识审核人员身份，关联朋友圈审核流程
属性	− auditorId : string − auditorName : string
方法	+getAuditor() : Auditor
关联关系	FriendCircle（审核对象）、FriendCircleController（审核流程触发）交互
输入来源	Controller 层调用）
输入内容	auditorId 字符串、auditorName 字符串；“审核请求事件”）

类名	Auditor
约束条件	1.auditorId 需全局唯一 2. auditorName 非空
输出目标	Controller 层（反馈审核员信息）、系统日志（可选，记录审核操作）
输出内容	Auditor 实例（包含审核员身份信息）
输出场景	
处理步骤关联	1.Auditor 实例并存入系统 2. 朋友圈审核时，Controller 调用关联逻辑 → 记录审核人信息

Model 层 - VerifiedUser（实体类）

类名	VerifiedUser
功能定位	实体模型，封装用户核心信息及社交功能操作方法（私信、朋友圈）
属性	- userName : string - userId : string
方法	+getUser(userId:string) : VerifiedUser +sendPrivateMessage(content:string, senderId:string, receiver:string) : boolean +publishFriendCircle(content:string, publisherId:string) : boolean
关联关系	PrivateMessage（私信发送者/接收者）、FriendCircle（朋友圈发布者）、PrivateMessageController（私信操作）、FriendCircleController（朋友圈操作）交互
输入来源	PrivateMessageUI、FriendCircleUI）的用户操作、系统用户管理模块
输入内容	userId 字符串、content 字符串、receiver 信息；“发送/发布请求事件”）
约束条件	1.userId 需全局唯一（注册时生成） 2. userName 非空
输出目标	Controller 层（反馈操作结果）、UI 层（反馈社交功能结果）
输出内容	boolean（操作是否成功） 结构化数据（如私信发送状态、朋友圈发布状态）
输出场景	
处理步骤关联	1.Controller 调用 sendPrivateMessage()→ 操作成功后反馈 UI 2. 发布朋友圈时，UI 层触发请求 → Controller 调用 publishFriendCircle()→ 审核通过后发布并反馈 UI

Model 层 - PrivateMessage（实体类）

类名	PrivateMessage
功能定位	实体模型，封装私信核心信息及基础操作方法
属性	- messageId : string

类名	PrivateMessage – content : string – sendTime : Date
方法	+savePrivateMessageHistory() : void +getPrivateMessageId() : string
关联关系	VerifiedUser（发送者/接收者）、PrivateMessageController（私信存储）、PrivateMessageUI（私信展示）交互
输入来源	Controller层（PrivateMessageController）的操作请求、系统时间（sendTime自动生成）
输入内容	messageId字符串、content字符串；“存储请求事件”）
约束条件	1.messageId需全局唯一（生成规则如 UUID+时间戳） 2. sendTime为发送时的系统时间（精度到毫秒）
输出目标	Controller层（反馈存储结果）、UI 层（反馈私信内容）
输出内容	void（存储完成无返回，或布尔值表示成功） messageId字符串（用于查询唯一私信）
输出场景	
处理步骤关联	Controller创建实例 → 调用 savePrivateMessageHistory()→ 持久化至messageStore然后加进数据库→ 反馈 UI 层

与其他正式交流数据库表

1.朋友圈表

字段名	数据类型	约束条件	备注
post_id	VARCHAR(64)	PRIMARY KEY, UNIQUE	UUID生成
publisher_id	VARCHAR(64)	FK(users.user_id)	发布者ID
content	TEXT	NOT NULL	动态内容
media_urls	JSON		附件URL列表（图片/视频）
visibility	ENUM	DEFAULT 'friends'	可见性('public','friends','private')
approved_at	TIMESTAMP		审核通过时间
is_deleted	BOOLEAN	DEFAULT FALSE	逻辑删除标记

2.私信表

字段名	数据类型	约束条件	备注
message_id	VARCHAR(64）	PRIMARY KEY, UNIQUE	UUID生成
sender_id	VARCHAR(64）	FK(users.user_id)	发送者ID

receiver_id	VARCHAR(64）	FK(users.user_id)	接收者 ID
content	TEXT	NOT NULL	私信内容
read_status	BOOLEAN	DEFAULT FALSE	是否已读
sent_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	发送时间
is_deleted	BOOLEAN	DEFAULT FALSE	逻辑删除标记

6.3.2 up 主用况

6.3.2.1 发布视频

1. 输入

输入内容	来源	数据类型
视频文件	up 主本地设备	二进制文件（视频格式）
视频封面文件	up 主本地设备	二进制文件（图片格式）
视频标题	up 主手动输入	字符串
视频简介	up 主手动输入	字符串

2.输出

输出内容	去向	数据类型
上传界面	up 主前端页面	可视化界面组件
上传进度反馈	up 主前端页面	进度条 / 百分比数值
错误提示信息	up 主前端页面	字符串
视频 URL	云数据库	字符串（URL 地址）
视频封面 URL	云数据库	字符串（URL 地址）
视频记录	云数据库	结构化数据（JSON / 数据表行）
粉丝新视频更新提醒	粉丝前端页面 / 消息中心	通知消息（文本 + 视频缩略信息）
审核结果反馈（通过 / 不通过）	up 主前端页面 / 消息中心	结构化消息（文本 + 原因说明）

6.3.2.2 管理视频

1.输入

输入内容	来源	数据类型	约束	合并/操作属性
up 主账户标识	系统当前登录态	字符串（用户 ID）	非空，且与视频所属 up 主 ID 一致	作为查询视频列表的核心筛选条件，无合并操作
视频选择指令	up 主前端操作	操作指令（关联视频 ID）	需选择当前 up 主已上传的有效视频（未被删除 / 隐藏）绑定	触发后续管理操作，与目标视频 ID
修改后的视频标	up 主手动输入	字符串	1. 非空；2. 长度	与其他修改内容

题			符合平台限制；3. 无违禁词汇	（封面、简介等）合并，覆盖原数据
修改后的视频封面	up 主本地设备 / 重新选择	二进制文件（图片格式）	1. 格式支持；2. 尺寸 / 分辨率符合平台要求；3. 无违禁内容	替换原封面文件，更新封面 URL 至云数据库
修改后的视频简介 / 关键词	up 主手动输入 / 选择	字符串 / 字符串数组	1. 长度符合平台限制；2. 无违禁词汇；3. 数量符合平台限制；4. 单个关键词长度限制；5. 无违禁词汇	合并至视频信息集合，覆盖原数据
评论管理操作	up 主前端操作	操作指令（关联评论 ID）	仅对当前视频的评论生效，操作权限仅限 up 主	单个评论独立操作，无合并逻辑，实时同步至数据库
视频删除 / 隐藏指令	up 主前端操作	操作指令	仅对当前 up 主所属视频生效，删除后不可恢复（或按平台规则处理）	触发视频状态更新（隐藏）或数据删除，无合并操作
重试指令（列表加载失败 / 操作失败后）	up 主前端操作	操作指令	仅在加载失败 / 操作失败场景下可用	触发流程恢复（重新加载列表 / 重新执行操作）

2.输出

输出内容	去向	数据类型	约束	合并/操作属性
视频列表（up 主已上传）	up 主前端页面	结构化列表数据（含视频 ID、标题、封面缩略图、上传时间、状态等）	1. 仅展示当前登录 up 主的视频；2. 数据与云数据库实时同步；3. 包含操作入口	1. 从云数据库查询并聚合视频核心信息生成；2. 加载失败时提示错误，支持重试
视频管理操作界面	up 主前端页面	可视化界面组件	与选择的视频绑定，仅展示该视频的可操作功能	无合并操作
操作结果提示（成功 / 失败）	up 主前端页面	字符串	1. 成功：明确告知“修改成功”“删除成功”等；2. 失败：包含具体原因	基于操作执行结果生成，不合并其他数据，触发流程回退（失败时）或刷新（成功时）
更新后的视频信息	云数据库	结构化数据	覆盖原视频记录	1. 合并所有修改内容生成；2. 审核通过后更新，审核不通过则恢复原数据；3. 与视频 URL 保持关

				联
刷新后的视频稿件列表	up 主前端页面	结构化列表数据	与“视频列表”结构一致，展示更新后的视频状态 / 信息	基于云数据库最新数据重新查询生成，覆盖原列表展示
审核结果反馈（修改后）	up 主前端页面 / 消息中心	结构化消息（文本 + 原因说明）	1. 审核通过：告知修改生效；2. 审核不通过：说明驳回原因	基于审核员操作结果生成，触发数据回滚（不通过时）或流程终止（通过时）

6.3.3 系统管理员用况

6.3.3.1 分析系统性能

输入来源	输入内容	数据类型	约束条件
系统管理员	登录监控平台请求	无结构化输入	仅触发流程，无数据校验
登录表单	管理员账号	String	非空、符合系统管理员账号格式
登录表单	密码	String	非空、长度 8-16 位、含数字与字母
监控平台交互	选择的层级（如用户体验、应用服务）	String	必须为系统预设的层级选项之一
监控平台交互	选择的日期（查看日志）	Date	格式为 YYYY-MM-DD，不可选择未来日期
监控平台交互	查看的指标类型（如 CPU、内存）	String	必须为系统支持的指标类型
监控平台交互	警告通知（来自系统）	JSON	包含警告 ID、级别、描述、时间戳

输出目标	输出内容	数据类型	输出场景
登录界面	登录表单展示	可视化界面	管理员发起登录请求后
监控仪表盘	性能数据展示	JSON + 图表	登录成功并选择层级后
监控仪表盘	警告列表展示	JSON	系统产生高优先级警告时
日志查看界面	指定日期日志内容	Text/JSON	管理员选择日期并确认查看权限后
问题定位页面	异常指标关联分析结果	JSON + 图表	管理员触发“调查资源瓶颈”后
系统记录	巡检记录	JSON	每次巡检结束时自动保存

处理步骤	处理逻辑	负责类 / 操作
1	接收管理员登录请求，展示登录表单	Sysadmin.loginRequest() + LoginPageUI.showForm()
2	校验账号密码，返回登录结果	AuthController.verifyAdmin(account, password)
3	登录成功，加载监控仪表盘	DashboardUI.loadDashboard(adminId)
4	获取并展示性能数据	PerformanceAnalysisController.fetchMetrics(layer)
5	检查指标是否正常	PerformanceMetric.isWithinThreshold()
6	记录巡检信息	PerformanceAnalysisController.recordInspection()

内存数据库 + 云服务操作

步骤	操作类型	具体操作
登录验证	云数据库读	从 `admin` 表查询账号与密码匹配记录
加载性能数据	云数据库读 + 缓存	从 `performance_metrics` 表读取指标数据，并存入 Redis 缓存
查看日志	云数据库读	从 `system_logs` 表按日期查询日志内容
记录警告	内存数据库写	将警告信息暂存入 `active_alerts` 队列
记录巡检	云数据库写	向 `inspection_logs` 表插入一条巡检记录
异常指标分析	内存数据库读	从缓存中读取关联指标数据进行关联分析

输出内容表

步骤	输出对象	输出内容
登录成功	系统 → 管理员	登录成功提示，跳转至监控仪表盘
登录失败	系统 → 管理员	“账号或密码错误”提示
加载性能数据	系统 → 管理员	层级性能面板（含图表与数值）
收到警告	系统 → 管理员	警告通知弹窗（含警告级别、描述、建议操作）
查看日志	系统 → 管理员	指定日期的日志内容列表
发现异常指标	系统 → 管理员	异常指标高亮显示，关联指标面板展开
巡检完成	系统 → 管理员	“巡检已记录”提示

数据库表结构（相关部分）

表名	字段名	数据类型	约束/备注
admin	admin_id	VARCHAR(32)	主键
admin	account	VARCHAR(32)	唯一、非空

表名	字段名	数据类型	约束/备注
admin	password	VARCHAR(64)	非空、加密存储
admin	role	VARCHAR(16)	默认：sysadmin
performance_metrics	metric_id	VARCHAR(64)	主键
performance_metrics	metric_type	VARCHAR(32)	如：cpu_usage, memory_usage, response_time
performance_metrics	value	FLOAT	当前值
performance_metrics	threshold	FLOAT	阈值
performance_metrics	timestamp	TIMESTAMP	记录时间
system_logs	log_id	VARCHAR(64)	主键
system_logs	log_level	VARCHAR(16)	INFO, WARN, ERROR
system_logs	content	TEXT	日志内容
system_logs	timestamp	TIMESTAMP	记录时间
inspection_logs	inspection_id	VARCHAR(64)	主键
inspection_logs	admin_id	VARCHAR(32)	外键：admin.admin_id
inspection_logs	start_time	TIMESTAMP	巡检开始时间
inspection_logs	end_time	TIMESTAMP	巡检结束时间
inspection_logs	status	VARCHAR(16)	COMPLETED, INTERRUPTED

6.3.4 系统管理员用况

6.3.4.1 更新系统

1.输入

输入来源	输入内容	数据类型	约束条件
系统管理员	登录监控平台请求	无结构化输入	仅触发流程，无数据校验
系统操作员	登录运维平台请求	无结构化输入	仅触发流程，无数据校验
登录表单	操作员账号	String	非空、符合操作员账号格式
登录表单	密码	String	非空、长度 8-16 位、含数字与字母
更新仪表盘交互	点击“启动更新流程”	事件	前置条件检查通过后可用
更新仪表盘交互	选择“服务更新”任务	事件	当前步骤为待执行状态时可用

服务更新交互	选择应用程序包	String (包 ID/名称)	必须为已上传且通过验证的包
服务更新交互	选择目标服务器集群	String (集群 ID)	必须为有效、健康的集群
服务更新交互	确认执行更新	事件	选择完成后可用
服务更新交互	执行“一键回滚”操作	事件	仅在监控阶段指标严重超标时出现
数据库更新交互	选择/上传数据库脚本	File / String (脚本 ID)	文件类型限定.sql，或为预置脚本
数据库更新交互	确认模拟报告并批准执行	事件	模拟运行完成且报告确认无误后可用
系统自动监控	关键性能指标数据	JSON	包含错误率、响应时间、资源使用率等
流程控制	点击“暂停更新”/“继续更新”	事件	在流程执行中任何步骤可用
系统自动检查	前置检查结果	JSON	包含健康检查、资源检查（磁盘、节点状态）的通过与否及详情
系统自动验证	更新包校验结果	JSON	包含数字签名有效性、完整性校验结果

2.输出

输出目标	输出内容	数据类型	输出场景
登录界面	登录表单展示	可视化界面	操作员发起登录请求后
运维平台主界面	平台主菜单与模块入口	可视化界面	登录成功后
系统更新仪表盘	更新流程总览面板	JSON + 可视化看板	进入“系统更新”模块后
系统更新仪表盘	前置检查结果面板	JSON + 状态指示	启动更新前或检查失败时
服务更新面板	待选应用程序包列表	JSON	点击“执行服务更新”后
服务更新面板	目标集群选择器及状态	JSON	进入服务更新子流程后
服务更新面板	滚动发布实时状态看板	JSON + 图表	发布流程启动后
数据库更新面板	脚本模拟运行影响报告	Text/JSON + 格式化文档	脚本模拟执行完成后
数据库更新面板	脚本执行进度与日志	Text/JSON	生产脚本执行启动后
监控与验证面板	关键性能指标实时图表	JSON + 时序图表	新服务节点开始接收流量后
监控与验证面板	高优先级警告通知弹窗	JSON	关键指标超过严重阈值时

更新仪表盘	流程步骤状态更新（完成/失败/暂停）	JSON + 状态指示	每个步骤执行结果产生后
操作反馈	包验证失败错误提示	可视化提示	更新包验证不通过时
操作反馈	回滚操作执行结果	可视化提示	回滚操作完成后

3.处理

处理步骤	处理逻辑	负责类 / 操作
1	接收操作员登录请求，校验账号密码	`AuthController.verifyOperator(account, password)`
2	登录成功，加载运维平台主界面，导航至“系统更新”模块	`OpsPlatformUI.loadDashboard(operatorId)` -> `UpdateModuleUI.show()`
3	启动更新流程前，执行系统健康与资源前置检查	`PreCheckController.executeSystemChecks()`
4	前置检查通过，初始化并展示更新流程仪表盘与步骤	`UpdateOrchestrator.initFlow()` -> `UpdateDashboardUI.render(flowSteps)`
5	操作员按顺序触发各步骤（如服务更新），执行业务逻辑	`UpdateOrchestrator.executeStep(stepName, params)`
6	在关键步骤（如服务发布）后，启动系统指标监控与告警	MonitoringController.startPostUpdateMonitoring(targets)`
7	若监控到严重异常，提供并执行回滚操作	RollbackHandler.triggerRollback(updateId)`
8	操作员确认更新完成，记录完整的更新流水日志	`UpdateOrchestrator.finalizeAndRecord(updateId, operatorId)`

4.内存数据库 + 云服务操作

步骤	操作类型	具体操作
登录验证	云数据库读	从 `operator` 表查询账号与密码匹配记录
加载更新配置	云数据库读 + 缓存写	从 `update_packages`, `server_clusters` 表读取数据，存入 Redis
前置检查	云服务 API 调用 + 内存数据库写	调用健康检查 API，结果暂存于 `precheck_results` 缓存键
执行服务更新	内存数据库写 + 云服务 API 调用	向 `deployment_queue` 写入任务；调用容器服务 API 进行发布
模拟数据库脚本	云数据库读/写（备份库）	在备份数据库实例上执行脚本模拟，结果写入 `script_simulation_reports` 表
执行数据库更新	云数据库写（生产库）	在生产数据库上执行已验证的脚本，日志写入 `db_update_logs` 表
监控指标	内存数据库写 + 时序数据库写	实时指标暂存于 Redis Sorted Set；聚合后写入时序数据库（如 InfluxDB）
触发回滚	内存数据库读/写 + 云服务 API	从 `rollback_scripts` 缓存读取回滚指令；调

调用		用负载均衡与数据库 API 执行
记录流程状态	内存数据库写 + 云数据库写	当前步骤状态更新到 `update_progress` 缓存键；最终日志写入 `system_update_logs` 表

5.输出内容表

步骤	输出对象	输出内容
登录成功	系统 → 操作员	登录成功提示，跳转至运维平台主界面
登录失败	系统 → 操作员	“账号或密码错误”提示
前置检查通过	系统 → 操作员	“检查通过，可开始更新”提示，并激活启动按钮
前置检查失败	系统 → 操作员	检查失败警告面板，列出具体失败项及修复建议
进入更新模块	系统 → 操作员	系统更新仪表盘，展示预设的更新步骤流程图
服务更新启动	系统 → 操作员	应用程序包列表、集群选择器及发布状态看板
包验证失败	系统 → 操作员	“更新包验证失败”错误提示，中止当前部署子流程
数据库脚本模拟完成	系统 → 操作员	模拟运行影响报告（涉及的表、行数预估、操作类型）
监控指标异常	系统 → 操作员	警告通知弹窗（含异常指标、级别、建议执行回滚）
回滚操作完成	系统 → 操作员	“回滚成功，系统已恢复”提示，并更新流程状态为“已回滚”
更新流程完成	系统 → 操作员	“所有更新步骤已完成，系统运行稳定”提示，更新记录已保存

6.数据库表结构（相关部分）

表名	字段名	数据类型	约束/备注
operator	operator_id	VARCHAR(32)	主键
	account	VARCHAR(32)	唯一、非空
	password	VARCHAR(64)	非空、加密存储
	role	VARCHAR(16)	默认：`sys_operator`
update_packages	package_id	VARCHAR(64)	主键

	name	VARCHAR(128)	更新包名称
	version	VARCHAR(32)	版本号
	type	VARCHAR(16)	`SERVICE`, `DB_SCRIPT`, `CONFIG`
	file_path	VARCHAR(256)	存储路径（云存储 URL）
	signature	VARCHAR(512)	数字签名，用于验证
	upload_time	TIMESTAMP	上传时间
	status	VARCHAR(16)	`UPLOADED`, `VERIFIED`, `INVALID`
server_clusters	cluster_id	VARCHAR(32)	主键
	name	VARCHAR(64)	集群名称
	environment	VARCHAR(16)	`PROD`, `STAGING`
	load_balancer_id	VARCHAR(64)	关联的负载均衡器
	current_version	VARCHAR(32)	当前运行版本
	health_status	VARCHAR(16)	`HEALTHY`, `UNHEALTHY`
db_update_scripts	script_id	VARCHAR(64)	主键
	name	VARCHAR(128)	脚本描述
	content_hash	VARCHAR(64)	脚本内容哈希值
	rollback_script_id	VARCHAR(64)	关联的回滚脚本 ID，外键
	risk_level	VARCHAR(16)	`LOW`, `MEDIUM`, `HIGH`
system_update_logs	update_id	VARCHAR(64)	主键
	operator_id	VARCHAR(32)	外键：`operator.operator_id`
	start_time	TIMESTAMP	更新流程开始时间

end_time	TIMESTAMP	更新流程结束时间
target_version	VARCHAR(32)	目标版本
final_status	VARCHAR(16)	`SUCCESS`, `ROLLBACK`, `FAILED`
log_details	TEXT	详细步骤执行日志（JSON 格式）

后记

参考文献

[1] YOUNG.RSS 是什么？[EB/OL]. http://jingpin.org/what-is-rss/.

[2] 杨博，彭博.RSS 提要分析与阅读器设计[R].成都：四川大学计算机学院，2007：42-43.

[3] 逸出络然.RSS 技术的原理[EB/OL].http://yclran.blog.163.com/blog/static/9794549620091110341115 58/.

[4] 佚名.Qt 是什么[EB/OL]. http://qt.nokia.com/title-cn.

[5] 佚名.Model/View Programming[EB/OL]. http://doc.trolltech.com/4.6/model-view-programming.html.

[6] [加拿大]Jasmin Blanchette[英]Mark Summerfield 著 闫锋欣,曾泉人,张志强译.

[7] C++ GUI Qt4 编程（第二版）[M].电子工业出版社：2008:182-206,291-305.

[8] 佚名.XML Processing[EB/OL]. http://doc.trolltech.com/4.6/xml-processing.html.

[9] Michael Blala James Rumbangh 著.UML 面向对象建模与设计（第 2 版）[M].北京：人民邮电出版社,2006:136-235.

[10] 胡海静,王育平,等. XML 技术精粹[M]. 北京：机械工业出版社，2001:17-19.